Privacy-aware Task Allocation and Data Aggregation in Fog-assisted Spatial Crowdsourcing

Haiqin Wu, Liangmin Wang, *Member, IEEE,* and Guoliang Xue, *Fellow, IEEE*

Abstract—Spatial crowdsourcing (SC) enables task owners (TOs) to outsource spatial-related tasks to a SC-server who engages mobile users in collecting sensing data at some specified locations with their mobile devices. Data aggregation, as a specific SC task, has drawn much attention in mining the potential value of the massive spatial crowdsensing data. However, the release of SC tasks and the execution of data aggregation may pose considerable threats to the privacy of TOs and mobile users, respectively. Besides, it is nontrivial for the SC-server to allocate numerous tasks efficiently and accurately to qualified mobile users, as the SC-server has no knowledge about the entire geographical user distribution. To tackle these issues, in this paper, we introduce a fog-assisted SC architecture, in which many fog nodes deployed in different regions can assist the SC-server to distribute tasks and aggregate data in a privacy-aware manner. Specifically, a privacy-aware task allocation and data aggregation scheme (PTAA) is proposed leveraging bilinear pairing and homomorphic encryption. PTAA supports representative aggregate statistics (e.g.,sum, mean, variance, and minimum) with efficient data update while providing strong privacy protection. Security analysis shows that PTAA can achieve the desirable security goals. Extensive experiments also demonstrate its feasibility and efficiency.

Index Terms—Spatial crowdsourcing, data aggregation, fog computing, bilinear pairing, homomorphic encryption.

1 Introduction

The proliferation of mobile devices, especially smartphones, has revolutionarily changed the traditional static sensor-based data sensing method to a new data collection and analysis paradigm named mobile crowdsensing (or crowdsensing for short). With advanced wireless communication techonologies (e.g., WiFi, 4G, and Bluetooth) and sensing capabilities of massive embedded sensors (e.g., camera, GPS, and accelerometer) on mobile devices, mobile users/participants can perform various tasks published on a crowdsourcing platform by the task owners (TOs) easily with their mobile devices. The sensing data are generated not only by the embedded sensors but also by the participant's personal knowledge, hence provide more comprehensive and diverse information for TOs. As a specific type of crowdsourcing [1], spatial crowdsourcing (SC) enables multiple TOs to outsource their spatio-temporal tasks to the SC-server that is responsible for recruiting a set of mobile users to complete tasks in the specified sensing location or area. Without deploying any static sensor in the specific areas, SC brings many benefits such as lower deployment cost, more sensing resources, and more sufficient spatial and temporal coverage. Moreover, the mobile participants can also earn rewards to compensate their consumption (e.g., battery, computing power, and 3G/4G quota) in participation.

Many promising SC applications are fostered in different domains such as environmental monitoring [2], assistive healthcare [3], intelligent transportation [4], and social survey [5]. In these SC applications, aggregating and analyzing the data submitted by mobile users are more useful and needed for TOs than simply collecting the raw sensing data, since many potential values can be explored through data aggregation. For example, the average rating of a certain restaurant will indicate its popularity among customers. The number of people having higher temperature than the normal in a city can help to supervise the flu spreading range and take related actions. The maximum movement speed of vehicles on a certain road during rush hours can provide real-time traffic information for path planning.

Despite the usefulness of SC, we observe that the privacy issue is a major hindrance towards the large-scale and successful deployment of SC applications, since the SC-server may not be fully trusted by both TOs and mobile users. Specifically, on one hand, tasks released by the TOs may inadvertently reveal their private information (e.g., points of interest (POI) and daily routine). Continuing with the aforementioned examples, if Alice, as a TO, requests the traffic information on a specific road, others may infer her presence nearby. Taking the restaurant rating for example, if someone requests the average rating of restaurants in a certain area,

- *H. Wu is with the Department of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China, 212013. Currently, she is a visiting PhD student in Arizona State University. E-mail: haiqinwu@asu.edu, whq92_ujs@126.com*
- *L. Wang is with the Department of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China, 212013. E-mail: wanglm@ujs.edu.cn*
- *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 the National Science Foundation grants 1421685, 1461886, 1704092, and 1717197, the National Natural Science Foundation of China under Grant U1736216, Grant 61472001, and Grant 61702233, and the National Key Research and Development Program Grant 2017YFB1400703. The information reported here does not reflect the position or the policy of the funding agencies. This work was done while H. Wu was visiting Arizona State University.

others may regard him/her interested in food or the catering industry, which may bring undesirable advertisements. Under these circumstances, it is necessary to keep the sensing tasks unknown to the SC-server. Moreover, the aggregation results are regarded as the private assets of the requested TOs who may not want to disclose these statistics to others. On the other hand, for the mobile participants, the tasks they want to join and the sensing data they submitted might both contain their private information like POI, location, and health status [6]. Mobile users may be reluctant to participate for fear of their privacy disclosure. Therefore, it is of paramount importance to preserve the privacy of both TOs and mobile participants in the SC process, including periods of task assignment and data aggregation.

There have been various attempts to address the privacy and security issues in crowdsourcing, which merely focus on the privacy protection in either task allocation ([7], [8], [9], [10]) or data collection and aggregation ([11], [12], [13], [14], [15], [16], [17]). Among this work, some researchers ([11], [12], [14]) assume the existence of one TO in the system and regard the TO and the SC-server as the same entity for simplicity, hence they ignore the privacy concerns of multiple TOs. Moreover, the aforementioned researches rely on a single server for both task distribution or data aggregation, which on one hand is hard to realize efficient and large-scale task allocation in different areas as some users cannot connect to the remote SC-server. On the other hand, significant overhead is incurred at the single SC-server, especially in the multiple TOs and multiple participants (MOMP) scenario. Although some aggregations are conducted by collaborations, either between the SC-server and the participants [18] or between neighboring participants [19], the extra communication cost and long delay make them less practical especially for the energy-constrained participants. Additionally, in [11], [12], only sum aggregation is supported, which does not cover all real applications.

To tackle these issues, in this paper, inspired by the salient benefits of fog computing [20] such as geographic distribution, location awareness, and low latency, we propose to take full advantage of the fog node to assist the SC-server to assign spatial tasks and aggregate the sensed data in a privacy-aware manner. The proposed scheme ensures that the SC-server can assign tasks to their corresponding fog nodes without knowing task contents. In addition, many aggregation statistics can be performed on encrypted data and no one is able to obtain the aggregated results except for the requested TOs. The main contributions of this paper are as follows:

- A privacy-aware task allocation and data aggregation scheme is proposed in the novel fog-assisted spatial crowdsourcing. We adopt the two-step task allocation (SC-server to fog nodes and fog nodes to participant) method, which offloads the heavy burden of the single SC-server and realizes large-scale task allocation. Meanwhile, apart from the aggregation results, the task content of each TO is well protected from the SC-server while the identity of each TO keeps confidential and cannot be linked to the task content.
- We adopt the oblivious transfer protocol to achieve privacy-aware task secret distribution without revealing which tasks the participants want to take. For privacy-preserving data aggregation, we elaborate some secure two-party aggregation protocols which support representative statistics such as sum, mean, variance, and minimum on encrypted data. Moreover, two efficient data update methods are devised regarding two different cases.
- To the best of knowledge, this is the first secure task allocation and data aggregation scheme to address the privacy issues of both TOs and participants in the MOMP and fog-assisted SC scenario. Security analysis and experimental results also demonstrate its security and efficiency.

The remainder of this paper is organized as follows. In Section 2, we review the related work. Section 3 introduces some preliminaries such as the system architecture, threat model, design goals, and cryptography primitives. Section 4 describes the proposed scheme in detail. The security analysis and performance evaluations are presented in Section 5. Finally, Section 6 concludes the paper.

2 RELATED WORK

Existing research efforts have been devoted to privacy protection issues on different phases of general mobile crowdsourcing, ranging from task allocation to data collection and data aggregation.

2.1 Privacy-aware task allocation

Various privacy protection techniques are utilized to meet the different privacy requirements of either TOs or participants during the task allocation phase in crowdsourcing environments. To protect the participant's location privacy when requesting tasks, obfuscation [21], perturbation ([8], [22]), and encryption-based approaches [23] were widely employed to hide or distort the accurate location of participants. For task preferences privacy, Shin et al. [24] proposed an anonymous network-based (e.g., Tor) privacy-preserving framework for both task assignment and data submission, which disassociates the participant with his/her interested/requested tasks. [25] achieved the protection of TO's task privacy and participant's privacy simultaneously by proposing a secure task matching scheme based on proxy re-encryption. Although the server cannot learn the specific keywords in which participants are interested, the identical ciphertext derived from the same keyword would indicate that a set of workers have the same task preference. To prevent the privacy disclosure from task contents, Ni et al. [9] protected the task confidentiality in two stages, which relies on the security of q-DBDHI assumption and AFGH proxy re-encryption, respectively. Recently, few works have studied the privacy-aware task assignment in the fog-assisted SC, in [10], an encryption-based method was adopted for task privacy, and enabled the assigned fog nodes and mobile users to decrypt the task. Their solution, however, is not secure enough as the task can also be decrypted by the curious SC-server.

2.2 Privacy-aware data aggregation

Generally, existing privacy-aware data aggregation schemes can be categorized into two groups: collection-oriented data aggregation which focuses on collecting high-quality sensed data, and statistic-oriented data aggregation which aims to conduct various statistic analysis on sensed data.

For collection-oriented data aggregation, Qiu et al. [19] proposed SLICER, which incorporates data coding technique and message transfer strategies to achieve k-anonymous privacy protection for multimedia data. In their design, the authors aim to reconstruct the original multimedia data while maintaining high data quality and achieving participants' privacy protection. To improve the quality of sensory data collected from multiple participants, the topic of truth discovery [26] has drawn wide attention recently. Accordingly, some researchers have proposed a series of privacy-preserving truth discovery frameworks ([27], [28], [29]) in order to find the truthful information of each sensing object through reliability-aware aggregation without compromising user privacy.

For statistic-oriented data aggregation, Shi et al. [18] proposed a privacy-preserving data aggregation scheme based on data slicing and mixing techniques, which supports additive aggregation and some non-additive (such as max/min) aggregation functions. This scheme hides the data of participants from the server by slicing data into many parts and distributing all but one of those slices among their neighbors. However, the collusion between neighbors and the server would inevitably reveal users' original data and this solution suffers from the single-point failure problem (i.e., the server cannot obtain correct results if any neighbor fails to submit data). Similar problems also exist in [30], [13]. Without involvements of neighbor participants, Li et al. [11] adopted a secure additive homomorphic stream cipher called CMT to provide efficient Sum and Min aggregation on encrypted data. Nevertheless, this solution relies on a trusted third party TTP for secrets distribution, and requires extra communications to deal with dynamic user join-in and leaving. Moreover, the security level would degrade when more users collude with the server. Recently, Fan et al. [12] proposed a novel privacy-aware and trustworthy Sum aggregation protocol for mobile sensing, which can protect the data privacy of benign users and detect invalid data submitted by malicious users, even if multiple users collude with each other. In the fog-computing environment, Lu et al. [31] employed the homomorphic Paillier encryption and Chinese remainder theorem to aggregate hybrid sensing data, and early filtered the false data from unauthenticated devices using one-way hash chain. However, these aggregation protocols are limited to one or two simple statistic computations, which is not enough for diverse statistic analysis. As an improvement, in [15], more aggregation computations (such as mean, variance) are supported over ciphertexts by using BGV homomorphic encryption. For a given task, the public key of the requested TO is sent to the server, which is then assigned to the users for data encryption. Obviously, both servers and users can easily link two or more tasks to the same TO via public key. From another perspective, instead of hiding data, [14] presented an anonymous data aggregation protocol that achieved n-source anonymity. Concretely, bitwise-XOR homomorphic encryption is adopted to recover all user data without knowing their sources. Although this scheme can support more aggregation operations in the plaintext, it is actually at the cost of data confidentiality disclosure.

More importantly, we observe that the aforementioned researches have two crucial limitations. On one hand, most work is based on the single-server model, which inevitably poses a heavy burden at the server and does not efficiently fit the MOMP SC scenario. On the other hand, the privacy of multiple TOs is not fully considered in the whole SC process, i.e., both in task allocation and data aggregation phases.

3 Preliminaries

In this section, we give an overview of our system model, as well as a description of the security model, design objective, and some cryptography primitives.

3.1 System model

As shown in Fig. 1, a fog-assisted spatial crowdsourcing system consists of a spatial crowdsourcing server (SC-server), multiple task owners (TOs), a set of fog nodes, multiple mobile participants, and an authority center (AC). The elements of the system are explained below.

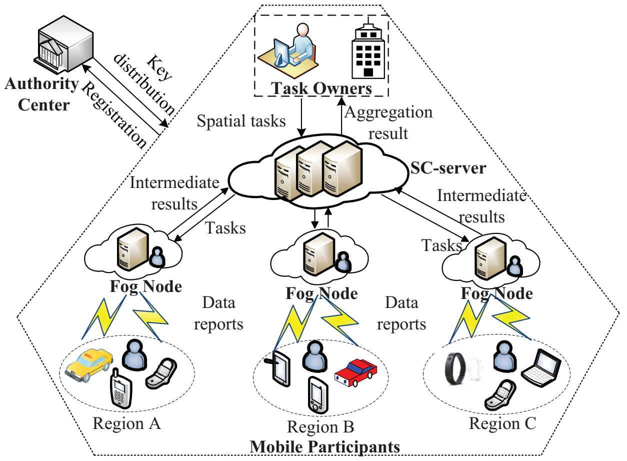

Fig. 1. System architecture

- Spatial crowdsourcing server (SC-server): The SC-server provides aggregation services for TOs by assigning their corresponding tasks to fog nodes according to the target area. Specifically, the SC-server receives the encrypted data from fog nodes and jointly computes the encrypted aggregated results with fog nodes. Finally, the encrypted results are returned to the TOs for decryption.
- Task owners (TOs): TOs can be individuals or organizations who generate spatial aggregation tasks and aim to obtain aggregation statistics over data collected from specific locations/areas of interests. However, it is nontrivial to accomplish these tasks individually due to their limited storage and computation capability. Hence, TOs release their tasks on

a SC-server along with some reward information (to incentivize users participation).

- Fog nodes: In fog-assisted crowdsourcing, the fog nodes are deployed at the edge of network and act as the relay between the SC-server and mobile participants. Particularly, each fog node manages a geographical area and is in charge of task distributions in its covering area. It collects encrypted sensing data from mobile participants and collaborates with the SC-server to accomplish some computations, which are then sent to the SC-server for further processing. Particularly, fog nodes are able to detect invalid (e.g., modified) data from adversaries.
- Mobile participants: Mobile participants or users[1] are those who carry mobile sensing devices and can flexibly perform sensing tasks according to their own willingness (e.g., if the given rewards are sufficient to compensate their sensing cost). To protect their data privacy, sensing data are encrypted before submitted to the fog nodes.
- Authority center (AC): The AC is responsible for bootstrapping the whole system, registering all system entities, and distributing corresponding keys to them. After that, it will be offline and does not participate in the crowdsourcing process.

3.2 Security model and assumptions

In our security model, the AC is considered fully trusted and will not be attacked by any adversary, since it manages all system parameters and keys. We assume that the TOs, the SC-server, fog nodes, and mobile participants are all honest-but-curious, which means that they will strictly conform to the designated protocols and honestly calculate the aggregated results, but try to infer more privacy of others, based on the information they hold.

Security threats come from two aspects: external and internal attacks. More specifically, external attackers may eavesdrop on communication channels to intercept the encrypted sensing data, intermediate and aggregated results. Moreover, they may disguise as legitimate and authorized participants to submit false data to the fog nodes. On the other hand, internal attackers could be any system entity except for the AC. For example, a task owner may want to learn the aggregated results of some tasks issued by other TOs. The mobile participants may also try to know other sensing data and TOs' aggregated results. The SC-server and fog nodes are both curious to know more sensitive information (e.g., identity, health status, and location) about TOs and participants by tracking their interested tasks or sensing data. We assume that there is no collusion between the SC-server and any fog node, which is a common assumption in threat models used in cryptographic protocols [32].

3.3 Design objective

Based on the aforementioned system and security models, we aim to design a privacy-aware task allocation and data aggregation scheme in fog-assisted SC. Particularly, we aim to achieve the following three objectives:

1. We use mobile users and participants interchangeably in this paper.

1) *Privacy preservation*. Our scheme should meet two aspects of privacy requirements as below.
   - **Privacy of TOs**. During the task allocation phase, the SC-server cannot learn the exact task content of TOs. Although it is inevitable for the SC-server to know the exact fog nodes to which tasks are sent, neither the SC-server nor a fog node is aware of the tasks assigned to a given participant. Conversely, the participants and fog nodes do not know the owner(s) of their assigned task(s). During the data aggregation phase, the aggregated results are not revealed to any entity except for the TO generating the aggregation task.
   - **Privacy of participants**. Neither the SC-server nor a fog node can infer if a given participant has been assigned specific task(s), or determine whether two tasks have been assigned to the same participant. Moreover, the sensing data of participants should be kept confidential from other entities.
2) *Security*. Unauthorized participants or external adversaries cannot forge or modify the sensing data without being detected.
3) *Efficiency*. The computation cost of our proposed scheme should be as less as possible at each entity, especially at the mobile users.

3.4 Cryptography primitives

1) *Bilinear Pairing* [33]: Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_\mathbb{T}$ be three multiplicative cyclic groups with the same prime order p, where p is κ bits, $\mathbb{G}_1$ and $\mathbb{G}_2$ are generated by g and h, respectively. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\mathbb{T}$ with the following properties:
   - Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and random numbers $a, b \in \mathbb{Z}_p^*$;
   - Non-degeneracy: There exists $g \in \mathbb{G}_1, h \in \mathbb{G}_2$ which satisfies $e(g, h) \neq 1$;
   - Computability: For all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, $e(g_1, g_2)$ can be computed efficiently.
2) *Oblivious Transfer* (OT) [34]: OT is a two-party protocol which enables a receiver to learn one or several input secrets of a sender while the sender does not know which secrets the receiver has learned. Moreover, apart from the requested secrets, the receiver cannot learn any information about other secrets. Particularly, in the 1-out-of-N OT (OT_1^N) protocol, the sender has N input strings $X_1, X_2, \ldots, X_N$ as secrets, the receiver can choose one secret X_I and does not learn any information about the other $N-1$ secrets. At the same time, the sender learns nothing about the secret index I (i.e., oblivious to which secret is transferred).
3) *Distributed Two Trapdoor Public-Key Cryptosystem* (DT-PKC) [35]: DT-PKC is a state-of-the-art cryptography primitive with additive homomorphic property which supports secure computations in the multi-key setting (i.e., multiple users). Compared

with the previous double trapdoor decryption cryptosystem [36], DT-PKC further avoids the risk of the strong trapdoor leakage by splitting the system's strong private key into two different shares. Hence, it provides higher security and more suitability for fog-assisted crowdsourcing. The main algorithms work as follows:

- **Setup**(1^k): Given a security parameter k, this algorithm outputs the system parameter $sp = (q_1, q_2, N, g')$ for DT-PKC, where q_1 and q_2 are two large prime numbers with k bits, $N = q_1 q_2$, and g' is a generator of order $(q_1 - 1)(q_2 - 1)/2$ [37].
- **KeyGen**(sp): Given the system parameter sp, we compute $SK = \lambda = lcm(q_1 - 1, q_2 - 1)/2$ as the system's strong private key, where lcm denotes the least common multiple of two numbers. For the weak private key of an entity i, we randomly select $sk_i = \theta_i \in [1, N/4]$ as his/her weak private key, and compute $h_i = g'^{\theta_i}\ mod\ N^2$, the corresponding public key is $pk_i = (N, g', h_i)$. In addition, we also define a function $L(x) = (x-1)/N$ for ease of presentation (to be used in data decryption).
- **SkeyS**(λ): This algorithm splits the system's strong private key $SK = \lambda$ into two partial strong private keys $SK^j = \lambda_j (j = 1, 2)$, where $\lambda_1 + \lambda_2 \equiv 0\ mod\ \lambda$ and $\lambda_1 + \lambda_2 \equiv 1\ mod\ N^2$.
- **Enc**(m, pk_i): Given a message $m \in \mathbb{Z}_N$ and a public key pk_i, this algorithm chooses a random number $r \in [1, N/4]$ and outputs the ciphertext $[m]_{pk_i} = (C_{i,1}, C_{i,2})$, where $C_{i,1} = g'^{r\theta_i}(1 + mN)\ mod\ N^2$ and $C_{i,2} = g'^{r}\ mod\ N^2$.
- **WDec**($[m]_{pk_i}, sk_i$): This algorithm performs decryption with the user's weak private key. Given a ciphertext $[m]_{pk_i}$ and a weak private key $sk_i = \theta_i$, it outputs $m = L((C_{i,1}/C_{i,2}^{\theta_i})\ mod\ N^2)$.
- **PSD1**($[m]_{pk_i}, \lambda_1$): This algorithm performs the first-step partial decryption with the partial strong private key λ_1. Given a ciphertext $[m]_{pk_i} = (C_{i,1}, C_{i,2})$ and the partial strong private key λ_1, it outputs the partial decrypted ciphertext $CT_i^{(1)} = (C_{i,1})^{\lambda_1} = g'^{r\theta_i\lambda_1}(1 + mN\lambda_1) mod\ N^2$.
- **PSD2**($[m]_{pk_i}, CT_i^{(1)}, \lambda_2$): This algorithm performs the second-step partial decryption with another partial strong private key λ_2. Given the partial decrypted ciphertext $CT_i^{(1)}$, it first executes $CT_i^{(2)} = (C_{i,1})^{\lambda_2} = g'^{r\theta_i\lambda_2}(1 + mN\lambda_2)\ mod\ N^2$), then computes $m = L(CT_i^{(1)} \cdot CT_i^{(2)})$.

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

For ease of presentation, the notations used in this paper are listed in Table 1.

TABLE 1
Notation Settings

Notations	Description
m	Number of TOs in the system
n	Number of participants in the system
d_i	The sensing data of participant p_i
λ_1, λ_2	Partial strong private keys
$\mathcal{L}(x)$	Bit-length of x
pk_x, sk_x	Key-pair (public and private keys) of entity x
$[m]_{pk_i}$	Ciphertext of message m encrypted by pk_i
pk_{Σ_i}	The random public key of task owner o_i

4 Our proposed scheme

In this section, we present our **P**rivacy-aware **T**ask **A**llocation and data **A**ggregation scheme (PTAA) for fog-assisted spatial crowdsourcing. PTAA consists of five phases: system initialization, task generation and allocation, data collection and aggregation, data decryption, and additional data update if necessary. For better understanding, we first give a brief overview of our proposed scheme.

4.1 Overview

In the system initialization phase, the AC generates the system parameters, registers the TOs, SC-server, fog nodes, and mobile participants by assigning them corresponding key pairs, respectively. To get the statistics of some spatial-related data with least privacy disclosure, each TO anonymously sends the encrypted task content and some essential information such as aggregation types and obfuscated sensing area to the SC-server. According to the obfuscated sensing area, the SC-server first assigns tasks to the corresponding fog nodes who can decrypt the task content and further publicize them to the local mobile participants[2]. Particularly, if a participant wants to take some sensing tasks, he/she interacts with the SC-server to obtain corresponding task secrets which are regarded as credentials for task authorization. Meanwhile, the SC-server cannot know which task secrets the participant requested. To protect the data privacy during the submission phase, each participant hides his/her sensing data with a random number and further encrypts them with his/her public key. For each encrypted data received, fog nodes first verify its authenticity (really from the authorized participants), and then collaboratively compute the encrypted aggregation with the SC-server. Finally, the SC-server returns the result to the corresponding TO who can decrypt and obtain the aggregated result. In addition, if a TO wants to timely update his/her statistic result, we present two solutions available to efficiently deal with data updates according to two different cases.

2. Local mobile participants mean those who are located in the coverage area of a fog node.

4.2 System initialization

This phase is run by the AC which consists of two steps. First, the AC generates necessary parameters and secrets for the system. We call this step *Setup*. Next, it registers all entities (including the TOs, SC-server, fog nodes, and mobile participants) into the system and assigns them the corresponding keys. We call this step *Entity registration and key distribution*.

1) Setup: Given the security parameters κ and k, the AC first generates the bilinear parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, p, G)$, where $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ are three multiplicative cyclic groups with the same prime order p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_{\mathbb{T}}$, g and h are the generators of $\mathbb{G}_1$ and$\mathbb{G}_2$, respectively, and $G = e(g, h)$. The AC randomly chooses $a \in \mathbb{Z}_p^*$ as its secret key and calculates its public key $A = h^a$. Next, the AC generates the DT-PKC parameters $sp = (q_1, q_2, N, g')$ by performing the algorithm **Setup**(1^k), as introduced in Section 3.4.

2) Entity registration and key distribution: For the SC-server, the AC will issue a pair of keys for task encryption and decryption during the registration phase. Specifically, the AC randomly chooses $s \in Z_p^*$ as the SC-server's secret key and computes $S = g^s$ as the public key. For each TO o_j and mobile participant p_i, the AC registers them by executing **KeyGen**(sp) and sending (sk_j^o, pk_j^o) and (sk_i^p, pk_i^p) to them, respectively. Note that, the system's strong private key $SK = \lambda$ is also split into λ_1 and λ_2 using algorithm **SkeyS**(λ) after this process. Regarding each fog nodes f_i, the AC randomly selects $\rho_i \in Z_p^*$ as the private key and computes its public key g^{ρ_i}. Moreover, the AC picks a random number $\alpha \in \mathbb{Z}_p^*$ and computes $A' = (g^{\alpha})^{1/a}$. α is sent to all fog nodes while A' is sent to the SC-server.

4.3 Task generation and allocation

After system initialization, in this phase, any TO with statistic task requirements will generate his/her task and outsource it to the SC-server. Then our PTAA performs two-step task allocation from the SC-server to the mobile users via a set of fog nodes, while the privacy of TOs and mobile users are well protected.

1) Task generation: Without loss of generality, we assume that there are m TOs in the system, denoted as $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$. If a TO $o_i \in \mathcal{O}$ wants to get the statistic information of some sensed data in a specific area, he/she first generates a spatial task $\tau_i = (\tau_i^c, \tau_i^o, \tau_i^l, \tau_i^t)$, where τ_i^c denotes the sensing content (e.g., temperature) of task τ_i, τ_i^o represents the aggregation/statistic operation (e.g., sum, average) to be performed, τ_i^l and τ_i^t denote the exact sensing location/area and expiration time, respectively. In addition, o_i also chooses a nonce $\varsigma_i \in [1, N/4]$ and computes a random public key $pk_{\Sigma_i} = (N, g, h_{\Sigma_i} = g^{(\theta_{o_i} + \varsigma_i)/2})$, which is used to re-encrypt the intermediate data during aggregation (to be described in Section 4.4). Note that the randomness of the public key prevents adversaries from inferring if two tasks are from the same TO. To protect the task privacy, o_i picks two random numbers r_1, $r_2 \in \mathbb{Z}_p^*$ and encrypts τ_i as follows.

$$c_1 = (\tau_i^c || \tau_i^l || \tau_i^t) e(S, h)^{r_1} G^{r_2},\ c_2 = h^{r_1},\ c_3 = A^{r_2}. \quad (1)$$

To enable the SC-server to assign tasks to the qualified fog nodes without any disclosure of the exact sensing location/area, τ_i^l is replaced by an obfuscated sensing area τ_i^r, indicating that by which fog node τ_i^l is covered. Finally, o_i sends the message $(c_1, c_2, c_3, \tau_i^o, \tau_i^r, pk_{\Sigma_i})$ to the SC-server. Note that τ_i^o is inevitably revealed to the SC-server for the implementation of aggregation analysis.

2) Privacy-aware task allocation: After receiving the above information, the SC-server assigns these tasks to the corresponding fog nodes who then further distribute them to the local participants in a privacy-preserving way. Without loss of generality, we assume that the SC-server holds a set of tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ which is identified with the corresponding indices and there is a set of fog nodes $\mathcal{F} = \{f_1, f_2, \ldots, f_k\}$ in the system. For any task $\tau_i \in \mathcal{T}$, the SC-server assigns τ_i to a set of qualified fog nodes, denoted as $\mathcal{F}_i$ that is located in τ_i^r. Concretely, it first calculates $c_1' = e(A', c_3)$, where A' is distributed by the AC before. Then, for each qualified fog node $f_j \in \mathcal{F}_i$, it computes the following formula:

$$\varphi_j = e(g^{\rho_j}, c_2)^s. \quad (2)$$

Obviously, we have $\varphi_j = e(g^{\rho_j}, h^{r_1})^s = G^{\rho_j \cdot r_1 \cdot s}$ according to the bilinearity property. Finally, the SC-server sends $(i, c_1, c_1', \varphi_j, pk_{\Sigma_i}, \tau_i^o)$ to f_j.

Upon reception, f_j performs decryption with its private key ρ_j and α, and gets the specific task requirements as follows:

$$\tau_i^c || \tau_i^l || \tau_i^t = \frac{c_1}{c_1'^{1/\alpha} \cdot \varphi_j^{1/\rho_j}}. \quad (3)$$

Subsequently, f_j publicizes the decrypted task requirements to its local participants and allocates them the corresponding tasks by issuing them task secrets. Task authorization is essential to thwart attacks from unauthorized participants. Suppose that there are η_j tasks $\tau_1, \tau_2 \ldots, \tau_{\eta_j}$ distributed to f_j, if a participant p_i located in the covering area of f_j wants to take some tasks, he/she first sends a task request message to f_j with his/her real identity. If f_j accepts the request[3], p_i will interact with f_j and obtain the desired task secrets without revealing which secrets are requested. Particularly, we adopt the OT protocol [34] which uses OT_1^2 as the primitive [38] to achieve privacy-aware secret acquisition. As shown in Algorithm 1, f_j first computes η_j commitment keys in the initialization phase (Line 1-6). Note that to further reduce the key size, a_j^0 can be set as 1 for all $j \in [1, \ell]$ without compromising the system security. In the transfer phase (Line 7-14), f_j and p_i need to jointly perform OT_1^2 protocol ℓ times, in which p_i should pick $a_j^{i_j} r_j$ if he/she wants to learn any X_I. Finally, p_i can reconstruct K_I and use it to open the corresponding commitment Y_I.

4.4 Data submission and aggregation

In this phase, the participants send their encrypted data to the fog node(s) along with some verification information. The fog node(s) will verify the data integrity and aggregate all data according to the specified operation requirement, while not knowing each participant's data.

3. The fog node can decide whether to accept the task request according to many factors such as the trust level of participants. This line of work is orthogonal to ours.

Algorithm 1: Privacy-aware Task Secret Acquisition

```
Input: $f_j$ has $\eta_j$ task secrets $X_1, X_2, \ldots, X_{\eta_j}$, where $\eta_j = 2^\ell$.
Output: $p_i$ obtains the desired task secret $X_I$.
1  Generate $\ell$ random key pairs $(a_1^0, a_1^1), \ldots, (a_\ell^0, a_\ell^1)$;
2  for all $I \in [1, \eta_j]$ do
3      $\langle i_1, i_2, \ldots, i_\ell \rangle \leftarrow$ the bits of $I$;
4      Compute $K_I = g^{\Pi_{j=1}^{\ell} a_j^{i_j}}$ and $Y_I = commit_{K_I}(X_I)$;
5  end
6  Sends the commitments $Y_1, Y_2, \ldots, Y_{\eta_j}$ to $p_i$;
7  if $p_i$ wants to participate in task $I$ then
8      $f_j$ chooses random elements $r_1, r_2, \ldots, r_\ell$;
9  end
10 for each $j \in [1, \ell]$ do
11     $OT_1^2[f_j, p_i] \overset{pick}{\rightarrow} a_j^{i_j} r_j$;
12 end
13 $f_j$ sends $g^{1/r_1 r_2 \cdots r_\ell}$ to $p_i$;
14 $p_i$ reconstructs $K_I = (g^{1/(r_1 r_2 \cdots r_\ell)})^{(a_1^{i_1} r_1) \cdots (a_\ell^{i_\ell} r_\ell)}$ and opens the commitment $Y_I$ and reveal $X_I$;
```

Assuming that p_i has obtained data d_i for a task τ_j, to protect the data confidentiality, p_i first chooses a random number r_i and encrypts d_i with his/her public key pk_i to get $\mu_i = [d_i + r_i]_{pk_i}$. Then, p_i needs to compute some verification information to allow the fog node to validate the data integrity. Specifically, p_i generates a hash message authentication code (HMAC) for μ_i under the task secret key X_j: h_i =HMAC$_{X_j}(j|\mu_i|r_i|t_i)$, where j is the identifier of task τ_j and t_i is the current time slot. Finally, p_i sends the message (j, μ_i, r_i, h_i) to the corresponding fog node $f_j \in \mathcal{F}_j$ with a randomly generated pseudonym.

When f_j receives all messages from the participants, it first verifies if the received data are really from authorized participants and are not tampered with by any adversary. Specifically, f_j finds the jth task secret X_j and computes h' =HMAC$_{X_j}(j|\mu_i|r_i|t_i)$ based on the current time slot. Then, it checks $h' \overset{?}{=} h_i$, if this equation holds, μ_i is considered authentic without being falsified. Otherwise, μ_i is considered invalid and will be rejected.

Subsequently, f_j starts to aggregate all valid data according to τ_j^o with the assistance of the SC-server. Concretely, we formally illustrate several aggregation protocols for some representative statistics such as computing sum, mean, variance, minimum and maximum.

Sum. Suppose that there are a set of participants $\mathcal{N} = \{p_1, p_2, \ldots, p_{n_j}\}$ submitting data to f_j for o_i's task τ_j. Algorithm 2 describes the secure sum aggregation protocol, which allows f_j to get the encrypted summation of all sensed data $[\sum_{p_i \in \mathcal{N}} d_i]_{pk_{\Sigma_i}}$.

Firstly, each encrypted data $\mu_i = [d_i + r_i]_{pk_i}$ is partially decrypted by f_j using λ_1, and the intermediate result μ_i' is sent to the SC-server along with μ_i. Then, the SC-server can decrypt μ_i' and get $\mu_i'' = d_i + r_i$ with λ_2, after which the summation of μ_i'' (denoted by S in Line 10) is computed for all $p_i \in \mathcal{N}$. The SC-server re-encrypts the summation S with the random public key of o_i and sends it to f_j. Upon receiving $[S]_{pk_{\Sigma_i}}$, f_j first calculates the summation of all

Algorithm 2: Secure Sum Aggregation (SSA)

```
Input: $f_j$ has $\lambda_1, pk_{\Sigma_i}, \bigcup_{p_i \in \mathcal{N}} r_i, \bigcup_{p_i \in \mathcal{N}} \mu_i$; SC-server has $\lambda_2, pk_{\Sigma_i}$.
Output: The encrypted summation of data $[\sum_{p_i \in \mathcal{N}} d_i]_{pk_{\Sigma_i}}$.
1  // Fog node $f_j$;
2  for all $p_i \in \mathcal{N}$ do
3      $\mu_i' \leftarrow PSD1(\mu_i, \lambda_1)$;
4      Send $(\mu_i, \mu_i')$ to the SC-server;
5  end
6  // SC-server;
7  for all $p_i \in \mathcal{N}$ do
8      $\mu_i'' \leftarrow PSD2(\mu_i, \mu_i', \lambda_2)$;
9  end
10 $S \leftarrow \sum_{p_i \in \mathcal{N}} \mu_i''$;
11 Compute $[S]_{pk_{\Sigma_i}}$;
12 Send $[S]_{pk_{\Sigma_i}}$ to $f_j$;
13 // Fog node $f_j$;
14 Compute $R \leftarrow \sum_{p_i \in \mathcal{N}} r_i$ and $[R]_{pk_{\Sigma_i}}$;
15 $[\sum_{p_i \in \mathcal{N}} d_i]_{pk_{\Sigma_i}} \leftarrow [S]_{pk_{\Sigma_i}} \cdot ([R]_{pk_{\Sigma_i}})^{N-1}$;
```

random numbers, which is also encrypted with pk_{Σ_i} (Line 14). Finally, according to the homomorphic property, f_j can remove the random numbers and obtain the encrypted summation of all sensed data (Line 15).

Algorithm 3: Secure Mean Aggregation (SMA)

```
Input: $f_j$ has $\lambda_1, pk_{\Sigma_i}, \bigcup_{p_i \in \mathcal{N}} r_i, \bigcup_{p_i \in \mathcal{N}} \mu_i$; SC-server has $\lambda_2, pk_{\Sigma_i}$.
Output: The encrypted approximate mean of data $[\bar{d}]_{pk_{\Sigma_i}}$.
1  // Fog node $f_j$ and SC-server;
2  $[D]_{pk_{\Sigma_i}} \leftarrow [\sum_{p_i \in \mathcal{N}} d_i]_{pk_{\Sigma_i}}$;
3  // Fog node $f_j$;
4  $x \leftarrow 1/|\mathcal{N}|$;
5  $x \leftarrow (m, e)$ such that $x = m \cdot \beta^e$;
6  Compute $[m]_{pk_{\Sigma_i}}, [e]_{pk_{\Sigma_i}} \leftarrow ([-e]_{pk_{\Sigma_i}})^{N-1}$;
7  // Fog node $f_j$ and SC-server;
8  $[m']_{pk_{\Sigma_i}} \leftarrow$ SM$([m]_{pk_{\Sigma_i}}, [D]_{pk_{\Sigma_i}})$;
9  $[\bar{d}]_{pk_{\Sigma_i}} \leftarrow ([m']_{pk_{\Sigma_i}}, [e]_{pk_{\Sigma_i}})$;
```

Mean. Intuitively, we can derive the mean easily via division in the plaintext domain when given the sum and the number of participants. However, it is hard to derive the encrypted mean given the encrypted sum and number because DT-PKC does not directly support division computation in the ciphertext domain. Although secure multiplication **SM** protocol in [39] is an alternative to compute the encrypted mean by converting the division operation to the multiplication operation, it cannot be applied directly since **SM** only supports computations on BigInteger. Inspired by the IEEE 754-2008 standard for the floating-point arithmetic, we convert the floating-point number $1/|\mathcal{N}|$ into two integers m and e, which represents its integral significand and exponent, respectively. As shown in Algorithm 3, $x = m \cdot 10^e$ and we then compute two ciphertexts $[m]_{pk_{\Sigma_i}}$ and $[e]_{pk_{\Sigma_i}}$ (Line 6). Note that $[e]_{pk_{\Sigma_i}}$ can be derived from its encrypted inverse $[-e]_{pk_{\Sigma_i}}$ since $e < 0$. Finally, the fog node and the SC-server jointly compute the encrypted integral significand of the mean value (i.e., $[m']_{pk_{\Sigma_i}}$) by running the **SM** protocol which outputs $[D \cdot m]_{pk_{\Sigma_i}}$. Correspondingly, the encrypted mean, denoted as $[\overline{d}]_{pk_{\Sigma_i}}$, can be represented by two encrypted integers $[m']_{pk_{\Sigma_i}}$ and $[e]_{pk_{\Sigma_i}}$. In this case, o_i can recover the mean value by computing $m' \cdot 10^e$ after decryption.

Algorithm 4: Secure Variance Aggregation (SVA)

```
**Input**: $f_j$ has $\lambda_1, pk_{\Sigma_i}, \bigcup_{p_i\in\mathcal{N}} r_i, \bigcup_{p_i\in\mathcal{N}} \mu_i, [\overline{d}]_{pk_{\Sigma_i}}$;
        SC-server has $\lambda_2, pk_{\Sigma_i}$.
**Output**: The encrypted approximate variance
        $[\sum_{p_i\in\mathcal{N}} (d_i - \overline{d})^2/\mathcal{N}]_{pk_{\Sigma_i}}$.
1  // Fog node $f_j$;
2  Choose a random number $r \in \mathbb{Z}_N$ and
   $r \geq max(\bigcup_{p_i\in\mathcal{N}} r_i)$, and compute $[r \cdot 10^{-e}]_{pk_{\Sigma_i}}$;
3  $M \leftarrow [m']_{pk_{\Sigma_i}} \cdot [r \cdot 10^{-e}]_{pk_{\Sigma_i}}$;
4  $M' \leftarrow PSD1(M, \lambda_1)$;
5  **for** *all* $p_i \in \mathcal{N}$ **do**
6      Compute $R_i = (r - r_i) \cdot 10^{-e}$;
7      $\mu_i' \leftarrow PSD1(\mu_i, \lambda_1)$;
8      Send $(M, M', \mu_i, \mu_i')$ to the SC-server;
9  **end**
10 // SC-server;
11 $M'' \leftarrow PSD2(M, M', \lambda_2)$;
12 **for** *all* $p_i \in \mathcal{N}$ **do**
13     $\mu_i'' \leftarrow PSD2(\mu_i, \mu_i', \lambda_2)$;
14     $X_i \leftarrow [M'' - \mu_i'' \cdot 10^{-e}]_{pk_{\Sigma_i}}$,
       $Y_i = [(M'' - \mu_i'' \cdot 10^{-e})^2]_{pk_{\Sigma_i}}$;
15     Send $X_i, Y_i$ to $f_j$;
16 **end**
17 // Fog node $f_j$;
18 **for** *all* $p_i \in \mathcal{N}$ **do**
19     $Z_i \leftarrow X_i \cdot [R_i]_{pk_{\Sigma_i}}^{N-1}$;
20     $Z_i' \leftarrow Y_i \cdot Z_i^{N-2R_i} \cdot [R_i^2]_{pk_{\Sigma_i}}^{N-1}$;
21 **end**
22 $Z \leftarrow \prod_{p_i\in\mathcal{N}} Z_i'$;
23 Execute Line 4-9 in SMA ;
24 $[\sum_{p_i\in\mathcal{N}} (d_i - \overline{d})^2/n_j]_{pk_{\Sigma_i}} \leftarrow ([m'']_{pk_{\Sigma_i}}, [3e]_{pk_{\Sigma_i}})$;
```

Variance. Algorithm 4 shows our secure variance aggregation protocol (SVA) which is based on the execution result of SMA. Given the encrypted average value $[\overline{d}]_{pk_{\Sigma_i}}$, the basic idea is to first compute the encrypted sum of square $[\sum_{p_i\in\mathcal{N}} (d_i - \overline{d})^2]_{pk_{\Sigma_i}}$, and then derive the encrypted variance leveraging the SMA protocol.

It is noted that $[\overline{d}]_{pk_{\Sigma_i}}$ is represented by $([m']_{pk_{\Sigma_i}}, [e]_{pk_{\Sigma_i}})$, in which m' is 10^{-e} times as large as the actual mean value and decides the result of variance. Hence, we focus on considering $[m']_{pk_{\Sigma_i}}$ in the variance computation. As presented in Algorithm 4, f_j initially hides the mean $\overline{d}$ by choosing a random number $r > max(\bigcup_{p_i\in\mathcal{N}} r_i)$ (to ensure that $R_i > 0$). For consistency, r is also increased to 10^{-e} times before encryption. After getting M, M and μ_i are both partially decrypted using λ_1, and the intermediate results are then sent to the SC-server. Using λ_2, the SC-server can further decrypt M', μ_i' and get $M'' = (\overline{d}+r) \cdot 10^{-e}$ and $\mu'' = d_i + r_i$, respectively. Subsequently, the SC-server encrypts $M'' - \mu'' \cdot 10^{-e}$ and $(M'' - \mu'' \cdot 10^{-e})^2$ with pk_{Σ_i}, after which the ciphertexts are sent to f_j. At the fog node, the random numbers are further removed and we get $Z_i' = [(m' - d_i \cdot 10^{-e})^2]_{pk_{\Sigma_i}}$ (Line 20). For each $p_i \in \mathcal{N}$, we can get the encrypted sum of squares by multiplication (Line 22). Finally, following Line 4-9 in SMA, the encrypted variance of all sensed data can be derived and represented by $([m'']_{pk_{\Sigma_i}}, [3e]_{pk_{\Sigma_i}})$, where m'' is 10^{-3e} times as large as the actual variance.

Minimum. To compute the minimum value of all sensed data, we employ a secure less than protocol **SLT** [35], which is used as the building block for our secure minimum aggregation protocol (SMinA). Given two encrypted data $[d_i]_{pk_i}$ and $[d_j]_{pk_j}$, **SLT** outputs an encrypted data $[u']_{pk_{\Sigma_i}}$ which indicates the relationship between d_i and d_j. Specifically, $u' = 0$ shows that $d_i \geq d_j$ while $u' = 1$ shows that $d_i < d_j$.

Algorithm 5: Secure Minimum Aggregation (SMinA)

```
**Input**: $f_j$ has $\lambda_1, pk_{\Sigma_i}, \bigcup_{p_i\in\mathcal{N}} r_i, \bigcup_{p_i\in\mathcal{N}} \mu_i$; SC-server has
        $\lambda_2, pk_{\Sigma_i}$.
**Output**: The encrypted minimum data
        $[min(\bigcup_{p_i\in\mathcal{N}} d_i)]_{pk_{\Sigma_i}}$.
1  // Fog node $f_j$;
2  **for** $i = 1$ *to* $|\mathcal{N}|$ **do**
3      $\mu_i' \leftarrow PSD1(\mu_i, \lambda_1)$;
4      Send $\mu_i, \mu_i'$ to the SC-server;
5  **end**
6  // SC-server;
7  **for** $i = 1$ *to* $|\mathcal{N}|$ **do**
8      $\mu_i'' \leftarrow PSD2(\mu_i, \mu_i', \lambda_2)$;
9      Compute $[\mu_i'']_{pk_{\Sigma_i}}$;
10     Send $[\mu_i'']_{pk_{\Sigma_i}}$ to $f_j$;
11 **end**
12 // Fog node $f_j$;
13 **for** $i = 1$ *to* $|\mathcal{N}|$ **do**
14     $[d_i]_{pk_{\Sigma_i}} \leftarrow [\mu_i'']_{pk_{\Sigma_i}} \cdot ([r_i]_{pk_{\Sigma_i}})^{N-1}$;
15 **end**
16 // Fog node $f_j$ and SC-server;
17 $x \leftarrow [d_1]_{pk\Sigma_i}$;
18 **for** $i = 2$ *to* $|\mathcal{N}|$ **do**
19     $[u']_{pk_{\Sigma_i}} \leftarrow$ **SLT**$(x, [d_i]_{pk\Sigma_i})$;
20     $Z_1 \leftarrow x \cdot ([d_i]_{pk_{\Sigma_i}})^{N-1}$;
21     $Z_2 \leftarrow$ **SM**$([u']_{pk_{\Sigma_i}}, Z_1)$;
22     $x \leftarrow [d_i]_{pk\Sigma_i} \cdot Z_2$;
23 **end**
```

Based on the **SLT** and **SM** protocols, we elaborate SMinA in detail, as shown in Algorithm 5. Given all encrypted hidden data $(\mu_1, \mu_2, \ldots, \mu_{n_j})$, f_j and the SC-server first jointly compute the original sensing data encrypted under pk_{Σ_i} (i.e., $[d_i]_{pk_{\Sigma_i}}$ in Line 14). Subsequently, we set the initial minimum as d_1 and compute its ciphertext $x = [d_1]_{pk_{\Sigma_i}}$. For the rest encrypted data, we iteratively compute the encrypted minimum (Line 18-23). Specifically, take the first iteration as an example, after executing **SLT** protocol, we compute $Z_1 = [d_1 - d_i]_{pk_{\Sigma_i}}$ and $Z_2 = [u' \cdot (d_1 - d_i)]_{pk_{\Sigma_i}}$ with **SM**. Finally, we set $x = [d_i + u' \cdot (d_1 - d_i)]_{pk_{\Sigma_i}}$ by performing ciphertext multiplication. It is obvious that if $d_1 < d_i$, we have $x = [d_1]_{pk_{\Sigma_i}}$ as the encrypted minimum since $u' = 1$ when the above inequality satisfies. Through $|\mathcal{N}| - 1$ iterations, the encrypted minimum can be derived. For secure maximum aggregation, we only need to make two modifications in Line 20 and Line 22: $Z_1 \leftarrow ([d_i]_{pk_{\Sigma_i}}) \cdot x^{N-1}$, $x \leftarrow x \cdot Z_2$.

4.5 Data decryption and result acquisition

After obtaining the encrypted aggregated results, each fog node $f_j \in \mathcal{F}_i$ first sends the result to the SC-server. If multiple fog nodes are involved in the task τ_j (i.e., $(|\mathcal{F}_i| > 1)$), the SC-server needs to further aggregate the encrypted result leveraging the additive homomorphic property. Otherwise, the SC-server directly returns the encrypted result to o_i.

Upon receiption, o_i can derive the final aggregated result by executing decryption algorithm **WDec**() with his/her private key θ_{o_i} and the nonce ς_i chosen in the task generation phase. It is noted that other TOs, except for o_i, cannot decrypt the ciphertext and obtain the correct result, as pk_{Σ_i} is only associated with the random public key of o_i.

4.6 Data update

In some cases, the TOs may need to update the aggregated results periodically in order to get the latest statistics. It is, however, much more costly especially for the small-portion data update if all phases are undergoing from scratch. In this section, we provide two alternative and efficient solutions for data update (refer to as PPTA-DU1 and PPTA-DU2, respectively) targeting at two different cases, depending on whether the participants keep joining his/her chosen tasks.

According to the task requirement, the fog nodes broadcast a data update request to their local participants. If a participant p_i is still willing to contribute data to task τ_j, in our PPTA-DU1, he/she needs to respond to the request only when new data is generated. Take the sum operation as an example, assuming that p_i has new data d_i^n and old data d_i^o, then he/she computes $\mu_i^u = [d_i^n + 2r_i]_{pk_i} \cdot ([d_i^o + r_i]_{pk_i})^{N-1}$, and sends μ_i^u (i.e., $[\Delta d + r_i]_{pk_i}$, where $\Delta d = d_i^n - d_i^o$) and r_i to the fog node. Following similar steps in Algorithm 2, the fog node can obtain $[\Delta d + r_i]_{pk_{\Sigma_i}}$, with which the updated sum can be derived by computing $[\sum_{p_i \in \mathcal{N}} d_i]_{pk_{\Sigma_i}} \cdot [\Delta d + r_i]_{pk_{\Sigma_i}} \cdot ([r_i]_{pk_{\Sigma_i}})^{N-1}$.

On the contrary, if p_i decides to leave the task and another participant p_j wants to join, in our PPTA-DU2, p_i replies a *"leave"* message attached with his/her previous encrypted data $\mu_i = [d_i + r_i]_{pk_i}$, while p_j replies a *"join"* message attached with his/her encrypted data $\mu_j = [d_j + r_j]_{pk_j}$. In addition, both r_i and r_j are also sent to the fog node for later computation. After the partial decryption at the fog node, the SC-server can get $d_i + r_i$ and $d_j + r_j$ by further partial decryption. Then, it computes $\Delta_d + \Delta_r$ and encrypts it with pk_{Σ_i}, where $\Delta_d = d_j - d_i$ and $\Delta_r = r_j - r_i$. Finally, to obtain the update result, the fog node needs to compute $[\sum_{p_i \in \mathcal{N}} d_i]_{pk_{\Sigma_i}} \cdot [\Delta_d + \Delta_r]_{pk_{\Sigma_i}} \cdot ([\Delta_r]_{pk_{\Sigma_i}})^{N-1}$.

5 Performance Evaluation

In this section, we first analyze how **PATT** can achieve the privacy and security goals given in Section 3.3, then we demonstrate the efficiency of **PATT** in terms of computation and communication cost through extensive experiments.

5.1 Security analysis

Privacy of TOs. In our scheme, we mainly consider three aspects for the privacy of TOs, including identity privacy, task privacy, and result privacy. First, besides hiding the real identities of TOs via pseudonyms, we use the random public key pk_{Σ_i} to prevent the SC-server from inferring if two tasks are generated by the same TO. Second, the task contents are encrypted by TOs under public keys of both AC (i.e., A) and the SC-server (i.e., S). Moreover, the specific sensing location/area τ_i^l is obfuscated with a larger area τ_i^r. Therefore, the SC-server cannot obtain the detailed task information except for the aggregation operation, which is inevitably disclosed to ensure the successful task completion. For the next task assignment, only the qualified fog nodes and participants in the obfuscated sensing area can get the specific task contents without knowing their creators (i.e., TOs). During the data aggregation phase, the aggregated results are encrypted with pk_{Σ_i}, ensuring that only o_i holding θ_{o_i} and the nonce ς_i can decrypt and obtain the final results.

Privacy of participants. During the task allocation phase, leveraging the OT protocol, the SC-server cannot learn which task(s) a specific participant wants to take. In other words, given two tasks, the SC-server cannot derive whether two task secrets are sent to the same participant. In the data submission phase, each participant encrypts his/her sensed data d_i by using a random number r_i and the public key pk_i, the different random numbers chosen by each participant in DT-PKC make the ciphertext different even for the same sensing data. Therefore, the fog node or other adversaries will not be able to derive d_i from μ_i and cannot identify whether two ciphertexts have the same plaintext. For the data aggregation, the fog node first performs partial decryption, while the SC-server can only get the hidden plaintext after further decryption. Based on the re-encrypted intermediate results, the fog node can obtain the encrypted aggregation result based on the homomorphic property of DT-PKC under the same public key. Since the SC-server and fog nodes are non-collusive, none of them can derive the sensing data with their owned information.

Data security. In our scheme, the participant who wants to join task j needs to first request the corresponding task secret X_j from the fog node. X_j is then used as the key of HMAC which concatenates the task number j, the encrypted data μ_j, the random number r_j, and the timestamp t_i together. After receiving the message $\{j, \mu_j, r_j, h_i\}$, the fog node checks if $\text{HMAC}_{X_j}(j|\mu_i|r_i|t_i) = h_i$, this equation holds only when the participant has X_j (i.e., authorized to task j) and the submitted data are not tampered with by adversaries. In this way, the data security is guaranteed.

5.2 Experiment

Simulation setup. We conducted the simulations in Java based on JPBC library and BigInteger Class, to implement bilinear pairing operation and DT-PKC cryptosystem, respectively. In particular, the Type A elliptic curve, defined as $y^2 = x^3 + x$ with $\kappa = 160$, is used for the privacy-aware task distribution. While the BigInteger Class in java development kit is used for secure statistic computations, where N is set to be 1024 bits to achieve 80-bit security levels [40]. Moreover, we adopted 160-bit HMAC-SHA1 to perform data verification and ensure the data integrity. In the simulations, we varied the number of TOs m from 10 to 50 with each TO generating one spatial task for simplicity. We assumed that there were 10 fog nodes deployed in the spatial crowdsourcing area, and each task was uniformly distributed to these fog nodes. On the other hand, we varied the number of participants from 10 to 100 with an increment of 10, and each participant randomly generated a sensing data distributed over [0,100]. To further demonstrate the efficiency of our data update method, the number of participants is set from 200 to 1000 with an increment of 200. All simulations were run on a laptop with Intel Core i7-4650U CPU (1.7 GHz) and 8 GB RAM.

The performance metrics include running time at four entities during the task allocation and data aggregation phases. For each metric, we first measured the running time of some cryptographic primitives, then evaluated the impact of m and n (i.e., n_j in this paper, we assume that the same number of participants join any task j for easy presentation) on above metrics. Note that, we did not take any existing solution as the baseline for comparison, as those approaches either fail to consider the TOs' privacy or fit for our MOMP scenario.

Simulation results. We demonstrate the efficiency of our proposed scheme from the following four performance metrics.

1) Basic cryptographic primitive performance: Table 2 shows the running time of some basic cryptographic primitives such as bilinear pairing, HMAC, and different encryption and decryption operations in our protocol. The result is averaged over 100 runs.

TABLE 2
The performance of basic cryptographic primitives (ms)

Oper.	time	Oper.	Time	Oper.	Time
Pairing	16	Exp	15	HMAC	0.8
Enc1	20	Enc2	32	PSD1	10
PSD2	35	Dec1	6	Dec2	38

Here, Exp represents the exponentiation operation on the ciphertext. Enc1 and Enc2 denote one-time task and data encryption at TO and participant, respectively. PSD1 and PSD2 denote the first and second partial decryption, respectively performed at the fog node and the SC-server in our protocol. Last, Dec1 denotes the final decryption of integer aggregation result, while Dec2 is for decryption operation of float-point number result such as mean and variance. As we observe, the results demonstrate that decrypting the encrypted float-point number is most costly, while data encryption at the participant and partial decryption at the SC-server show comparable overhead. In contrast, HMAC is much more efficient with the lowest computation cost.

2) Cost of task allocation: In this paper, we mainly evaluate the computation time of the first-step task allocation (SC-server to fog nodes), as the cost in the second-step allocation (fog nodes to participants) depends on the OT protocol which has been analyzed in [34]. Fig. 2 shows the impact of m on the computation time at the SC-server and one qualified fog node, respectively. We observe that the running time of both SC-server and fog node increases with m. This is obvious due to the fact that the SC-server and the fog node need to perform proportional encryptions and decryptions with more tasks. Comparing the SC-server to the fog node, we note that the SC-server takes more time than the fog node. This is as expected, because the SC-server needs to perform more calculations than the fog node. For each task with k qualified fog nodes, the SC-server conducts k+1 pairing and k exponentiation operations, while each fog node mainly conducts two exponentiation operations when decrypting a task.

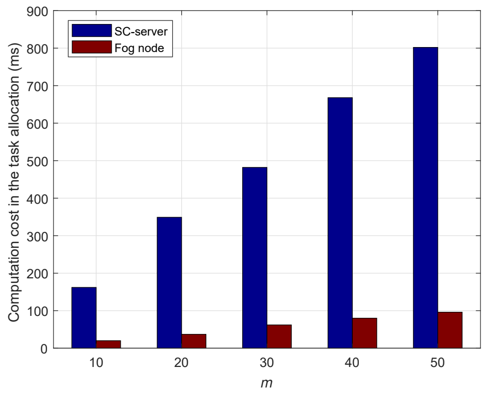

Fig. 2. The computation cost in the task allocation phase

3) Cost of data encryption: To demonstrate the user side efficiency during data submission and aggregation phase, we show in Fig. 3 the cost of data encryption at each user with respect to the number of tasks each user takes during data submission, as users are not involved in any computation of data aggregation. As we can see, the computation cost of data encryption linearly grows when each user takes more tasks. This is obvious since more data needed to be encrypted. When there are 50 tasks, the running time is about 1600ms, which is consistent with the result (Enc2) tested in Table 2.

3) Cost of data aggregation: Fig. 4 plots the running time of our four statistic aggregation protocols, respectively. To evaluate the efficiency of our secure aggregation scheme, we mainly compare the computation cost with varying n at the SC-server and the fog node who are two leading roles in our protocol execution.

Cost of sum aggregation. As we observe in Fig. 4(a), the result of the impact of n is consistent with the result in Fig.2, and the computation cost on both sides linearly increases as n grows. However, the SC-server's computation cost is more costly and grows more rapidly than the fog node, despite of the same number of partial decryption and encryption. This is because each PSD2 at the SC-server takes much more time (about three times as shown in Table (2) than each PSD1 at the fog node. Thus the second partial decryption dominates

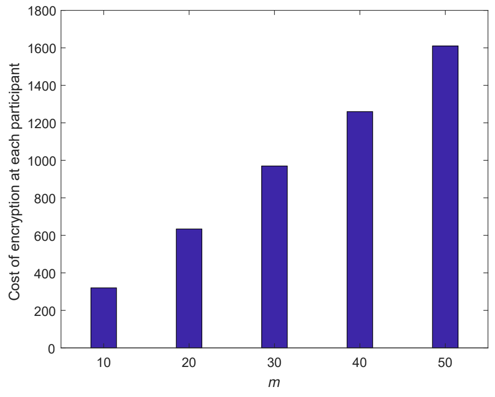

Fig. 3. The computation cost in the task allocation phase

the SC-server's cost and is nearly three times as much as that at the fog node. It is observed that when $n = 100$, the computation costs of the SC-server and the fog node are 3300ms and 1200ms, respectively.

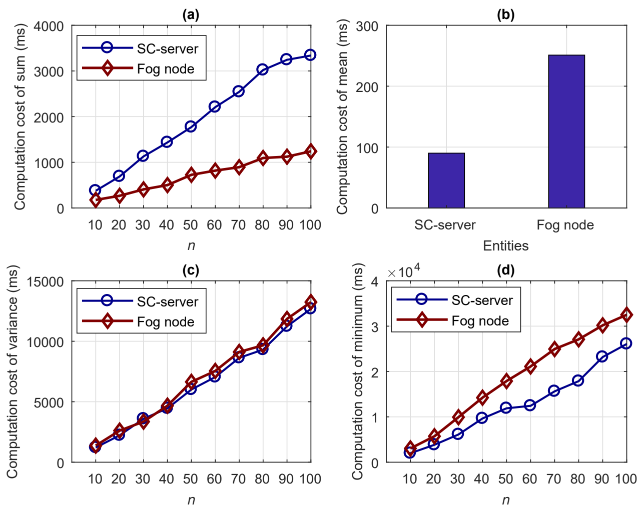

Fig. 4. The computation cost of four statistic aggregations

Cost of mean aggregation. As illustrated in Algorithm 3, our mean computation is based on the result of sum aggregation, it is observed that the impact of n on the cost of mean depends on its impact on the cost of sum aggregation. Therefore, we omit the impact of n and show the cost of mean aggregation at two entities when *given the encrypted sum and* n. As shown in Fig. 4(b), the fog node takes more running time (about 250ms) than the SC-server (about 90ms) when computing the encrypted average value (Algorithm 3), the reason is that **SM** protocol requires more computations at the fog node [39], moreover, the fog node first needs to convert the float-point number $1/|\mathcal{N}|$ into two encrypted BigInteger $[m]_{pk_{\Sigma_i}}$ and $[e]_{pk_{\Sigma_i}}$, which induces extra computation cost.

Cost of variance aggregation. Fig. 4(c) shows the impact of n on the cost of variance computation, which is consistent with the results in Fig. 3(a). Interestingly, we observe that the fog node shows slightly more running time than the SC-server, although more cost is incurred at the SC-server for PSD2. The reason is that the fog node requires to perform extra $3n$ exponentiation operations (each accounts for about 50% running time of PSD2) on the ciphertexts (Line 18-21, Algorithm 4), hence offsets the more computation cost of PSD2. Moreover, as shown in Fig. 4(b), SMA induces more cost at the fog node. Therefore, finally, the fog node and the SC-server present comparable computation cost. When $n = 100$, the computation cost at the SC-server is only 500ms more than that at the fog node.

Cost of minimum aggregation. Regarding the computation cost of minimum aggregation, as shown in Fig. 4(d), one straightforward observation is that the computation cost at the SC-server and the fog node both grows with increasing n, which is in accord with the aforementioned results. However, another observation is that the fog node incurs much more computation cost than the SC-server. This is reasonable due to the fact that the fog node needs another ($2n$-1) exponentiation operations (Line 14 and Line 20, Algorithm 5). In addition, both sub-protocols **SLT** and **SM** require the fog node to perform more computations than the SC-server. When $n = 100$, it takes 6000ms more than the SC-server, which are 32000ms and 26000ms at the fog node and the SC-server, respectively.

4) Cost of data updates: Finally, we evaluate the performance of our scheme in dealing with data updates. For better demonstration of our efficiency, we assume that there are 1000 participants in the system and simulate the computation cost with varying number of participants who really need to update their data. Take sum aggregation as an example, Fig. 5 compares the computation cost for data updates in the first case (i.e., some participants generate new data and no participants leave or join the system) with our PTAA-DU1 and without PTAA-DU1 (i.e., re-executing all operations), respectively. We observe that the total update cost at participant (Fig. 5(a)) in our scheme grows as an increasing amount of new data is generated, while the same cost is induced regardless of the updated participants without our update method. When the number of updated participants is more than 800, using PTAA-DU1 incurs more computation cost. This is because for n' updated participants, our update method requires n'(Enc2+Exp) computations, whereas re-execution requires nEnc2 computations ($n' \leq n$). At the fog node and the SC-server, PTAA-DU1 exhibits better performance with no expensive computations induced in sum aggregation (Fig. 5(b), 5(c)). As a result, from the perspective of the whole update process, our solution still significantly reduces the overall computation cost (about 37.7% when all old data are updated), as presented in Fig. 5(d).

Similarly, Fig. 5 shows the total computation cost for data updates in the second case (some participants leave and others join the system) with PTAA-DU2 and without PTAA-DU2, respectively. It is clear to see that the computation costs at three entities (Fig. 6(a)$\sim$(c)) are much less using our PTAA-DU2 than re-computations, especially when the number of newly joined participants is small. With the increased number of new (old) participants joining (leaving) the system, the computation cost in our scheme all increases and reaches the highest when all participants are updated, which is the same as the running time of re-execution. This is consistent with the fact that our scheme requires n'Enc2 computations for n' newly joined participants,

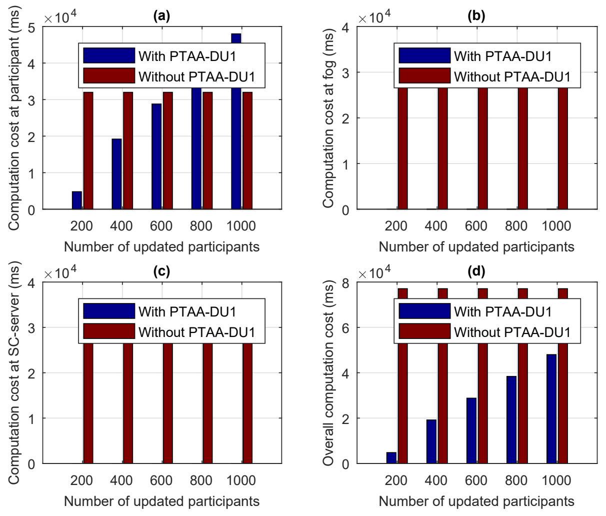

Fig. 5. The computation cost of data update (case 1)

while re-execution requires nEnc2 computations ($n' \leq n$). Meanwhile, in PTAA-DU2, extra n'PSD1+Enc2+Exp and n'PSD2+Enc2 computations are performed at the fog node and the SC-server, respectively. In contrast, it takes nPSD1+Enc2+Exp and nPSD2+Enc2 computations to re-execute the SSA protocol. As a whole, our update solution exhibits the superiority in the total running time of data update, which is also demonstrated in Fig. 6(d).

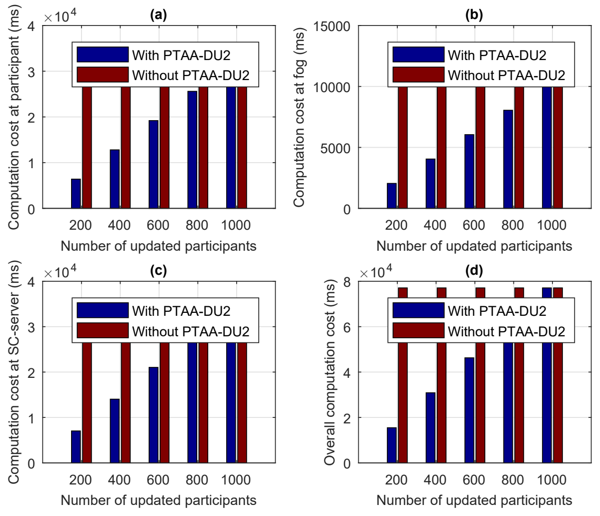

Fig. 6. The computation cost of data update (case 2)

Generally, in our **PTAA** scheme, the time costs incurred at the TO and the participant are much lower than that at the SC-server and the fog node, which achieves the lightweight participation especially for energy-constrained mobile users. To further accelerate the execution of our protocol and reduce the computation cost, some existing techniques, such as parallel computing [41] and packing method [42], can be employed seamlessly to our design.

6 Conclusion

In this paper, we propose a privacy-aware task allocation and data aggregation scheme **PTAA** for fog-assisted spatial crowdsourcing. **PTAA** adopts two-step task allocation strategy, which enables the SC-server to allocate tasks to qualified fog nodes without knowing the task content, and enables the fog nodes to further allocate tasks to qualified mobile participants without knowing if two tasks are from the same TO or two tasks are assigned to the same user. In the data aggregation phase, secure aggregation protocols allow the fog node(s) and the SC-server to jointly compute the encrypted aggregated result while ensuring the data and result privacy of participants and TOs, respectively. Simulation results show the scalability and efficiency of our proposed scheme.

References

[1] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: crowdsourcing with smartphones," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1732–1744, 2016.
[2] "irain2," http://irain.eng.uci.edu, [Online available].
[3] A. Ara, M. Al-Rodhaan, T. Yuan, and A. Al-Dhelaan, "A secure privacy-preserving data aggregation scheme based on bilinear elgamal cryptosystem for remote health monitoring systems," *IEEE Access*, vol. 5, pp. 12 601–12 617, 2017.
[4] "waze," http://waze.com, [Online available].
[5] "yelp," https://www.yelp.com/, [Online available].
[6] Y. Yang, X. Liu, R. H. Deng, and Y. Li, "Lightweight sharable and traceable secure mobile health system," *IEEE Transactions on Dependable & Secure Computing*, vol. PP, no. 99, pp. 1–14, 2017.
[7] X. Fan, P. Yang, Q. Li, D. Liu, Y. Zhao, and Y. Zhao, "Safe-crowd: secure task allocation for collaborative mobile social network," *Security & Communication Networks*, vol. 9, no. 15, pp. 2686–2695, 2016.
[8] H. To, G. Ghinita, L. Fan, and C. Shahabi, "Differentially private location protection for worker datasets in spatial crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 934–949, 2017.
[9] J. Ni, K. Zhang, X. Lin, Q. Xia, and X. S. Shen, "Privacy-preserving mobile crowdsensing for located-based applications," in *IEEE International Conference on Communications*, 2017, pp. 1–6.
[10] J. Ni, X. Lin, K. Zhang, and Y. Yu, "Secure and deduplicated spatial crowdsourcing: A fog-based approach," in *Global Communications Conference*, 2017, pp. 1–6.
[11] Q. Li and G. Cao, "Efficient and privacy-preserving data aggregation in mobile sensing," in *IEEE International Conference on Network Protocols*, 2012, pp. 1–10.
[12] J. Fan, Q. Li, and G. Cao, "Privacy-aware and trustworthy data aggregation in mobile sensing," in *Communications and Network Security*, 2015, pp. 31–39.
[13] L. Zhang, X. Wang, J. Lu, P. Li, and Z. Cai, "An efficient privacy preserving data aggregation approach for mobile sensing," *Security & Communication Networks*, vol. 9, no. 16, pp. 3844–3853, 2016.
[14] Y. Zhang, Q. Chen, and S. Zhong, "Privacy-preserving data aggregation in mobile phone sensing," *IEEE Transactions on Information Forensics & Security*, vol. 11, no. 5, pp. 980–992, 2016.
[15] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, "Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing," in *INFOCOM 2016 - the IEEE International Conference on Computer Communications, IEEE*, 2016, pp. 1–9.
[16] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Inception:incentivizing privacy-preserving data aggregation for mobile crowd sensing systems," pp. 341–350, 2016.
[17] Y. Sei and A. Ohsuga, "Differential private data collection and analysis based on randomized multiple dummies for untrusted mobile crowdsensing," *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 4, pp. 926–939, 2017.
[18] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "Prisense: privacy-preserving data aggregation in people-centric urban sensing systems," in *IEEE INFOCOM*, 2010, pp. 1–9.
[19] F. Qiu, F. Wu, and G. Chen, "Privacy and quality preserving multimedia data aggregation for participatory sensing systems," *IEEE Transactions on Mobile Computing*, vol. 14, no. 6, pp. 1287–1300, 2015.

[20] L. Rodero-Merino and L. Rodero-Merino, *Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing*. ACM, 2014.

[21] K. Vu, R. Zheng, and J. Gao, “Efficient algorithms for k-anonymous location privacy in participatory sensing,” in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 2399–2407.

[22] G. Ghinita, G. Ghinita, and C. Shahabi, *A framework for protecting worker location privacy in spatial crowdsourcing*. VLDB Endowment, 2014.

[23] Y. Shen, L. Huang, L. Li, and X. Lu, “Towards preserving worker location privacy in spatial crowdsourcing,” in *IEEE Global Communications Conference*, 2015, pp. 1–6.

[24] M. Shin, C. Cornelius, P. Dan, A. Kapadia, D. Kotz, and N. Triandopoulos, “Anonysense: A system for anonymous opportunistic sensing,” *Pervasive & Mobile Computing*, vol. 7, no. 1, pp. 16–30, 2011.

[25] J. Shu and X. Jia, “Secure task recommendation in crowdsourcing,” in *Global Communications Conference*, 2017, pp. 1–6.

[26] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, “Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation,” in *ACM SIGMOD International Conference on Management of Data*, 2014, pp. 1187–1198.

[27] 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 *ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 183–196.

[28] C. Miao, L. Su, W. Jiang, Y. Li, and M. Tian, “A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems,” in *INFOCOM 2017 - IEEE Conference on Computer Communications, IEEE*, 2017, pp. 1–9.

[29] Y. Zheng, H. Duan, and C. Wang, “Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing,” *IEEE Transactions on Information Forensics & Security*, vol. 13, no. 10, pp. 2475–2489, 2018.

[30] S. M. Erfani, S. Karunasekera, C. Leckie, and U. Parampalli, “Privacy-preserving data aggregation in participatory sensing networks,” in *IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013, pp. 165–170.

[31] R. Lu, K. Heung, A. Lashkari, and A. Ghorbani, “A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot,” *IEEE Access*, vol. 5, pp. 3302–3312, 2017.

[32] X. Liu, B. Qin, R. H. Deng, and Y. Li, “An efficient privacy-preserving outsourced computation over public data,” *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 756–770, 2017.

[33] E. J. Goh, *Encryption schemes from bilinear maps*. Stanford University, 2007.

[34] M. Naor and B. Pinkas, “Oblivious transfer with adaptive queries,” *Lecture Notes in Computer Science*, vol. 1666, pp. 573–590, 1999.

[35] X. Liu, R. H. Deng, K. K. R. Choo, and J. Weng, “An efficient privacy-preserving outsourced calculation toolkit with multiple keys,” *IEEE Transactions on Information Forensics & Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[36] E. Bresson, D. Catalano, and D. Pointcheval, “A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications.” in *Advances in Cryptology - ASIACRYPT 2003, International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, 2003, pp. 37–54.

[37] R. Cramer and V. Shoup, “Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption,” in *International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, 2002, pp. 45–64.

[38] M. Bellare and S. Micali, *Non-Interactive Oblivious Transfer and Applications*. Springer New York, 1989.

[39] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, “Secure k-nearest neighbor query over encrypted data in outsourced environments,” pp. 664–675, 2013.

[40] E. B. Barker, W. C. Barker, W. E. Burr, M. E. Smid, and M. E. Smid, “Recommendation for key management, part 1: General (revised),” *NIST Special Publication,*, vol. 800, no. 57, pp. 1–142, 2007.

[41] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing*. MIT Press, 1995.

[42] A. Liu, Z. Kai, L. Lu, and G. Liu, “Efficient secure similarity computation on encrypted trajectory data,” in *IEEE International Conference on Data Engineering*, 2015, pp. 66–77.

Haiqin Wu received the bachelor's degree in computer science from Jiangsu University in June 2014. She is working towards the PhD degree in the Department of Computer Science and Engineering, Jiangsu University. She is currently a visiting PhD student in the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. Her research interests include data security and privacy, mobile crowdsensing, and data query processing.

Liangmin Wang received his B.S. degree (1999) in Computational Mathematics in Jilin University, Changchun, China, and the PhD degree (2007) in Cryptology from Xidian University, Xi'an, China. He is a full professor in the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. He has been honored as a “Wan-Jiang Scholar” of Anhui Province since Nov. 2013. Now his research interests include data security & privacy.

He has published over 60 technical papers at premium international journals and conferences, like IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Vehicular Technology, IEEE Global Communications Conference, IEEE Wireless Communications and Networking Conference. Dr WANG has severed as a TPC member of many IEEE conferences, such as IEEE ICC, IEEE HPCC, IEEE TrustCOM. Now he is an associate editor of Security and Communication Networks, a member of IEEE, ACM, and a senior member of Chinese Computer Federation.

Guoliang Xue is a professor of Computer Science and Engineering at Arizona State University. He received the PhD 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, smart city and smart grids. He has published over 280 papers in these areas, many of which 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 Mobile Computing. 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 many conferences, including ACM CCS, ACM MOBIHOC, IEEE ICNP, and IEEE INFOCOM. He served on the editorial board of IEEE/ACM Transactions on Networking. He serves as the Area Editor of IEEE Transactions on Wireless Communications, overseeing 13 editors in the Wireless Networking area. He is an IEEE Fellow.