A Fog-Aided Privacy-Preserving Truth Discovery Framework over Crowdsensed Data Streams

Shaoxian Yuan, Bin Zhu, Feng Liu, Jian Li, Kaiping Xue†
School of Cyber Security, University of Science and Technology of China, Hefei, Anhui 230027, China
†Corresponding author, kpxue@ustc.edu.cn

***Abstract*—With the proliferation of mobile and wearable devices, mobile crowdsensing (MCS) is becoming a new paradigm for data collection and analysis. To effectively identify truthful information from crowdsensed data without privacy leakage, privacy-preserving truth discovery (PPTD) has gained much attention recently. Existing works either didn't consider real-time applications over data streams or failed to achieve enough efficiency for a large group of workers. In this paper, we propose FPTD, a Fog-aided Privacy-preserving Truth Discovery framework which is secure and efficient in handling real-time applications with a large group of workers. To reduce overhead, we adopt cloud-fog computing architecture to divide the complete worker group into many smaller ones. Then we design a unique secure aggregation protocol *SecAgg* which can securely and efficiently aggregate inputs from workers in smaller groups. Finally, we give detailed construction of FPTD, an efficient truth discovery framework based on *SecAgg* for real-time applications. Through extensive experiments and security analysis, we demonstrate that both *SecAgg* and FPTD are secure and efficient.**

***Index Terms*—fog computing, privacy-preserving truth discovery, mobile crowdsensing**

I. Introduction

With the rapid advancement and adoption of mobile, wearable and IoT devices, mobile crowdsensing (MCS) is becoming a new paradigm for data collection and analysis. Due to its low deployment cost, wide geographic coverage and flexibility, MCS has enabled a broad range of real-time applications, such as real-time traffic monitoring [1], urban dynamic mining [2]. In practice, data collected from mobile workers are noisy and unreliable due to background noise and manufacturing quality. In order to identify truthful information from noisy data, many truth discovery algorithms have been proposed [3], [4]. Despite the effectiveness, applying truth discovery to mine truthful information needs to tackle the privacy challenge. In many crowdsensing applications, data records provided by workers may contain sensitive information.

To tackle the privacy challenge, several privacy-preserving truth discovery (PPTD) schemes [5]–[8] have been proposed. Generally, they require workers to submit encrypted or masked data records to cloud servers and then perform truth discovery algorithm on encrypted or masked data. However, all these schemes incur limitations in handling real-time applications.

In particular, we address three challenges when designing an efficient PPTD scheme for real-time applications. First, data are collected as streams rather than static datasets in real-time applications. This means that each worker senses and uploads data records at regular intervals. On the one hand, this challenge requires a streaming truth discovery algorithm underlying PPTD scheme. On the other hand, it requires that the PPTD scheme should be able to report an estimation at regular intervals. Second, workers in real-time applications are dynamic. Workers in real-time applications may drop out at any time due to various reasons such as disconnection of network, unexpected termination of mobile device ,etc. This challenge requires the PPTD scheme to be failure robust, i.e., output correct estimation even if some workers drop out. Third, the number of workers may vary in scale. For large-scale real-time applications like dynamic mining, any mobile user could be a worker to contribute his/her social recording to earn some profits. This requires that PPTD scheme should be efficient even if the number of workers grows dramatically.

Roughly speaking, the majority of existing works [5]–[7] only focus on static data. Other work such as [8] is able to overcome the first two challenges while overlooking the third one, resulting in inefficiency with respect to a large group of workers. To be specific, the inefficiency of [8] is caused by the underlying secure aggregation protocol, called double masking [9]. For a large group of workers, double masking requires each worker to maintain a connection with every other worker to ensure privacy preservation. However, we argue that each worker maintaining connections with a much smaller group is enough to ensure privacy.

In particular, our contributions can be summarized below:

- First, we turn to cloud-fog computing architecture to divide the complete worker group into many smaller ones. In this way, we require each worker maintain connections with only its neighbours in the same group. This significantly reduces the number of connections each worker has to maintain.
- Second, we design a novel failure-robust secure aggregation protocol ***SecAgg*** which can securely aggregate inputs from all small worker groups efficiently.
- Finally, we give a detailed construction of FPTD framework based on ***SecAgg*** for real-time applications in MCS. Moreover, through extensive experiments and security analysis, we demonstrate that our FPTD is secure and efficient for real-time applications in MCS.

The remainder of this paper is organized as follows. In section II, we review existing works of PPTD. The system

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

model and threat assumption are given in section III. In section IV, we describe the underlying truth discovery algorithm and cryptographic primitives we adopt. In section V, we introduce the proposed scheme in detail. In section VI, we give security proof. Afterwards, performance evaluation is given in section VII. Finally, we conclude our paper in section VIII.

II. Related Work

The majority of existing works [5]–[7] focus on PPTD over static data. The first work of PPTD was presented by [5], which adopted threshold Paillier cryptosystem under a single cloud server. However, [5] introduced heavy computation burdens over both workers and cloud server. To optimize the performance, [6] utilized an improved Paillier cryptosystem and super-increasing sequence under the cloud-fog computing architecture. [7] realized a PPTD with an incentive mechanism to resist potential lazy workers by reducing the reward a lazy worker can obtain. However, existing PPTD work on static data cannot naturally extend to real-time applications due to the heavy cryptographic tools they adopt.

To address real-time applications, [8] adopted a lightweight secure aggregation protocol to realize PPTD which can support worker's dropout. Besides, [8] exploited a steaming truth discovery to efficiently identify truths from data streams. These two unique designs helps [8] overcome the first two challenges while overlooking the third one. Though some more lightweight aggregation protocol [10] based on differential privacy can help overcome the third challenge, the security would be degraded.

III. Problem Statement

A. System Model

Our system is a cloud-fog computing architecture. It consists of three entities: cloud server, fog nodes and workers. The cloud server is a remote server provided by a cloud service provider. Fog nodes are at the edge of network and close to workers. Workers can be devices with different sensors, such as smart phones, wearable and IoT devices. Cloud server maintains a connection with fog nodes and each fog node manages a group of workers. Fog nodes relay the messages between any two workers or workers and the cloud server. Formally, we assume there are M objects to be sensed $\mathcal{M} = \{o_1, \cdots, o_M\}$ and K fog nodes $\mathcal{F} = \{f_1, \cdots, f_K\}$. Each fog node f_u manages n_u workers $\mathcal{S}_u = \{s_1, s_2, \cdots, s_{n_u}\}$. In each time slot t, each worker collects data record $\boldsymbol{x}_i^t = \{x_i^{m,t}\}_{m=1}^M$, updates distance d_i^t and weight w_i^t. Estimated truth is represented as $\boldsymbol{x}_t^* = \{x_{m,t}^*\}_{m=1}^M$.

B. Threat Model

Similar to other PPTD works, we consider a semi-honest adversary where both fog nodes and cloud server faithfully behave as the protocol requires, while they try to infer private information from transcripts in the meantime. We also assume workers are honest because they are stimulated by incentives [11] which is another important topic in MCS.

In this paper, we build the security of ***SecAgg*** on the common assumption that no fog nodes will collude with the cloud server.

C. Design Goals

Our design goals are three folds.

Failure-robustness. The proposed FPTD framework should output correct estimation based on the remaining workers' sensory data even if some workers may drop out unexpectedly like [8].

Efficiency. The proposed FPTD should reach significantly lower overheads for both workers and cloud server compared with [8].

Security. We require the proposed FPTD can securely perform the process of truth discovery to identify truths from workers' noisy sensory data without revealing any data and weights.

IV. Preliminaries

In this section, we will illustrate the underlying truth discovery algorithm and cryptographic primitives.

A. Truth Discovery

The underlying truth discovery algorithm we adopt is the incremental CRH (iCRH) [3]. The basic idea behind iCRH is to assign a $weight$ to each worker to measure reliability and aggregate weighted average to obtain estimated $truth$.

The iCRH handles data streams. In each time slot t, iCRH is divided into three steps: truth update, distance update and weight update.

- Truth update: In this step, given each worker's sensory data $\boldsymbol{x}_i^t$ and weight w_i^{t-1} of last time slot, estimation of truth is updated as a weighted average

$$\boldsymbol{x}_t^* = \frac{\sum_{i=1}^S w_i^{t-1} \times \boldsymbol{x}_i^t}{\sum_{i=1}^S w_i^{t-1}}. \tag{1}$$

- Distance update: In this step, each worker updates their incremental distances d_i^t as below, given estimated truth, sensory data and previous distance d_i^{t-1}

$$d_i^t = \alpha \times d_i^{t-1} + d(\boldsymbol{x}_i^t, \boldsymbol{x}_t^*), \tag{2}$$

  where distance function $d(\boldsymbol{x}, \boldsymbol{y})$ measures the deviation level of two sensory data records of dimension M. For continuous data, we have $d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{m=1}^M (x_i^{m,t} - y_i^{m,t})^2$ and for categorial data, $d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{m=1}^M \mathbf{1}(x_i^{m,t}, y_i^{m,t})$ where $\mathbf{1}(x, y) = 1$ if $x \neq y$ and 0 otherwise. Besides, α represents decay rate to allow more recent data to play more important role.
- Weight update: In this step, given each worker's distance, each worker's weight is updated as

$$w_i^t = \log(\sum_{i=1}^S d_i^t) - \log(d_i^t). \tag{3}$$

B. Key Agreement

The key agreement establishes secrets between workers. It consists of three algorithms: $\mathbf{KA.param}$,$\mathbf{KA.gen}$ and $\mathbf{KA.agree}$.

- $\mathbf{KA.param}(1^\lambda) \mapsto (\mathbb{G}, g, p)$: Taking security parameter λ as input, the algorithm outputs public parameters pp.
- $\mathbf{KA.gen}(pp) \mapsto (c_i^{PK}, c_i^{SK})$: Taking the input of pp, the algorithm returns a public key pair to any worker i.
- $\mathbf{KA.agree}(c_i^{SK}, c_j^{PK}) \mapsto sc_{i,j}$: Given the secret key of s_i and the public key of s_j, the algorithm returns a shared key.

For correctness, we require that $\forall i \neq j$, $\mathbf{KA.agree}(c_i^{SK}, c_j^{PK}) = \mathbf{KA.agree}(c_j^{SK}, c_i^{PK}) = sc_{ij}$. For security, we rely on Decisional Diffle-Hellman assumption (DDH) [12].

C. Secret Share

We rely on Shamir's t-out-of-n secret share [13], which allows someone to split a secret s into n shares, such that any t shares can be used to reconstruct s, but any set of at most $t-1$ shares gives no information about s.

Specifically, sharing algorithm $\mathbf{SS.share}(s, t, \mathcal{U}) \mapsto \{(i, s_i)\}_{i\in\mathcal{U}}$ takes as input a secret s, a set $\mathcal{U}$ of n workers, and a threshold $t \leq |\mathcal{U}|$. For correctness, we require $\forall s, t, n$ with $1 \leq t \leq n$, $\forall |\mathcal{U}| = n$ if $\mathbf{SS.share}(s, t, \mathcal{U}) \mapsto \{(i, s_i)\}_{i\in\mathcal{U}}$, for $\mathcal{V} \subseteq \mathcal{U}$ and $|\mathcal{V}| \geq t$, then $\mathbf{SS.rec}(\{j, s_j\}_{j\in\mathcal{V}}, t) = s$.

V. Proposed Scheme

A. Overview

Double masking is a failure-robust secure aggregation protocol under the single server model. It employs two layers of masks to protect workers' inputs. Generally, the first layer is a zero-sum mask which can be canceled if no worker drops out. The second layer is a random mask which can only be removed by recovering the seeds. To handle workers' dropouts, double masking exploits t-out-of-n Shamir's secret sharing to distribute the secrets among all workers, which can help the central server to recover the two masks. Trivially applying double masking within each small group poses privacy breach since trivial solution not only outputs the final sum of inputs from all workers but also reveals partial sums of each group, which leaks extra information to a potential adversary.

We transform double masking to ***SecAgg*** via a re-mask technique. Our idea consists of two steps. First, similar to the trivial solution, we require each worker to add two layers of masks. However, we let each fog node remove the zero-sum mask and leave the random mask to the cloud server. Besides, we require each fog node adds one layer of the zero-sum mask to the intermediate value to preserve privacy against the cloud server.

After transforming double masking to ***SecAgg***, we construct the secure truth update and secure weight update for FPTD. Our construction is quite simple and straight. We utilize ***SecAgg*** to aggregate weights and weighted truths to update truths according to equation 1. Similarly, we aggregate distance information from all workers to update weights according to equation 3.

B. *SecAgg*

We observe that even though workers are dynamic and may drop out at any time, fog nodes are relatively stable and can maintain a long-term connection with the cloud server. Therefore we can avoid secret sharing among fog nodes which saves computation and communication.

Assume $n = \sum_{u=1}^{U} n_u$ represents the number of all workers and worker's id $i \in [n]$. Let S, s_i and f_u represent the cloud server, the u-th fog node and worker i respectively. Then we define $G(i, j) = 1$ if s_i and s_j are in same $\mathcal{S}_u$ managed by f_u, 0 otherwise. In particular, $\forall\, i \in [n], G(i, S) = 1$ holds. To support worker's dropout, the threshold for each small group is defined $t_u = \lceil \frac{3}{4} n_u \rceil$. We also assume each worker holds a vector $\boldsymbol{x}_i$ of dimension M'.

Round 0: One Time Initialization. In this round, ***SecAgg*** establishes long-term keys between workers and the cloud server. First, S generates public parameters pp which is a cyclic group $\mathbb{G}$ with prime order p and generator g via

$$\mathbf{KA.param}(1^\lambda) \mapsto (\mathbb{G}, p, g).$$

Then S publishes pp to all fog nodes and workers. After obtaining pp, each s_i including S generates a key pair

$$\mathbf{KA.gen}(pp) \mapsto (c_i^{PK}, c_i^{SK}),$$

where $i \in [n] \cup \{S\}$. After key generation, each s_i exchanges (i, c_i^{PK}) with every other s_j if $G(i, j) = 1$. Besides, S exchanges (S, c_S^{PK}) with every s_i. After the exchange, each s_i establishes key

$$sk_{i,j} = \mathbf{KA.agree}(c_i^{SK}, c_j^{PK}),$$

where $j \in [n] \cup S/\{i\}$ and $G(i, j) = 1$. Likewise, S establishes keys $sk_{i,S}\ \forall\ i \in [n]$. These keys serve as the keys for symmetric Authenticated Encryption, i.e. **AE**.

Round 1: Establish Zero-Sum Seeds. After one time initialization, both workers and fog nodes need to establish shared seed to generating zero-sum masks. Specifically, each s_i and f_u generates new key pairs

$$\mathbf{KA.gen}(pp) \mapsto (s_i^{PK}, s_i^{SK}),$$
$$\mathbf{KA.gen}(pp) \mapsto (z_u^{PK}, z_u^{SK}),$$

where $i, j \in [n]$ and $u, v \in [K]$. Then each s_i exchanges (i, s_i^{PK}) with every s_j where $G(i, j) = 1$ and each fog node exchanges (u, z_u^{PK}) with every f_v. Afterwards, each s_i and f_u establishes shared secrets

$$s_{i,j} = \mathbf{KA.agree}(s_i^{SK}, s_j^{PK}),$$
$$z_{u,v} = \mathbf{KA.agree}(z_u^{SK}, z_v^{PK}),$$

respectively.

Round 2: Share Secrets. In this round, each s_i shares his/her random seed and secret key within his/her group. Besides,

each fog node f_u shares his/her secret key with all other fog nodes. First, each s_i first uniformly samples random seed r_i for random masks. Then each s_i divides s_i^{SK}, r_i into shares via

$$\begin{aligned}\textbf{SS.share}(s_i^{SK}, t_u) \mapsto &\quad \{(j, s_{i,j}^{SK})\}_{j\in\mathcal{S}_u/\{i\}},\\ \textbf{SS.share}(r_i, t_u) \mapsto &\quad \{(j, r_{i,j})\}_{j\in\mathcal{S}_u/\{i\}},\end{aligned}$$

where $G(i,j)=1$. After dividing shares, each s_i encrypts the shares

$$e_{i,j} = \textbf{AE.encrypt}(sk_{i,j}, i||j||s_{i,j}^{SK}||r_{i,j}),$$

then each s_i exchanges $e_{i,j}$ with s_j where $j \in [n]$ and $G(i,j)=1$.

Round 3: Masked Input Collection. In this round, each s_i generates random mask $\boldsymbol{b}_i$ and zero-sum mask $\boldsymbol{s}_{i,j}$. For each $m \in [M']$, s_i computes each element

$$\begin{aligned}b_i^m =& H(r_i||m) \mod p,\\ s_{i,j}^m =& H(s_{i,j}||m) \mod p,\end{aligned}$$

where H is a cryptographic hash function. Then s_i adds masks to input

$$\begin{aligned}\boldsymbol{y}_i = \boldsymbol{x}_i + \boldsymbol{b}_i &\\ + &\sum_{j\in\mathcal{S}_u/\{i\}:i>j} \boldsymbol{s}_{i,j}\\ - &\sum_{j\in\mathcal{S}_u/\{i\}:i<j} \boldsymbol{s}_{i,j} \mod p.\end{aligned}$$

and sends masked input $\boldsymbol{y}_i$ to corresponding fog node f_u.

Round 4: Re-Mask. Assume some workers drop out before sending the masked inputs and we use $\mathcal{S}'_u$ to represent the surviving workers. Upon receiving masked inputs from surviving workers, f_u aggregates

$$\begin{aligned}\sum_{i\in\mathcal{S}'_u}\boldsymbol{y}_i &= \sum_{i\in\mathcal{S}'_u}\left(\boldsymbol{x}_i+\boldsymbol{b}_i + \sum_{j\in\mathcal{S}_u/\{i\}:i>j}\boldsymbol{s}_{i,j} - \sum_{j\in\mathcal{S}_u/\{i\}:i<j}\boldsymbol{s}_{i,j}\right)\\ &= \sum_{i\in\mathcal{S}'_u}(\boldsymbol{x}_i+\boldsymbol{b}_i) + \sum_{i\in\mathcal{S}_u}\sum_{j\in\mathcal{S}_u/\mathcal{S}'_u:i>j}\boldsymbol{s}_{i,j}\\ &\quad - \sum_{i\in\mathcal{S}_u}\sum_{j\in\mathcal{S}_u/\mathcal{S}'_u:i<j}\boldsymbol{s}_{i,j}.\end{aligned} \tag{4}$$

To recover $\sum_{i\in\mathcal{S}'_u}(\boldsymbol{x}_i+\boldsymbol{b}_i)$, each fog node f_u sends $\mathcal{S}'_u$ to surviving workers and request $\{s_{i,j}^{SK}\}_{i\in\mathcal{S}_u/\mathcal{S}'_u}$ from workers in $\mathcal{S}'_u$. After receiving shares from surviving workers, each fog recovers the secret key via **SS.rec**. Then f_u recovers the missing zero-sum masks and adds them to equation 4 to obtain the randomly masked input.

Afterwards, each f_u generates the new zero-sum mask. For each $m \in [M']$, f_u computes

$$z_{u,v}^m = H(z_{u,v}||m) \mod p,$$

then f_u computes

$$\boldsymbol{z}_u = \sum_{i\in\mathcal{S}'_u}(\boldsymbol{x}_i+\boldsymbol{b}_i) + \sum_{v\in\mathcal{F}:u>v}\boldsymbol{z}_{u,v} - \sum_{v\in\mathcal{F}:u<v}\boldsymbol{z}_{u,v}.$$

Finally, f_u sends z_u to cloud server.

Round 5: Unmasking. After receiving $\boldsymbol{z}_u$ from all fog nodes, S computes

$$\begin{aligned}C_{agg} &= \sum_{u\in\mathcal{F}}\left(\sum_{i\in\mathcal{S}'_u}(\boldsymbol{x}_i+\boldsymbol{b}_i) + \sum_{v\in\mathcal{F}:u>v}\boldsymbol{z}_{u,v} - \sum_{v\in\mathcal{F}:u<v}\boldsymbol{z}_{u,v}\right)\\ &= \sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{S}'_u}(\boldsymbol{x}_i+\boldsymbol{b}_i) + \sum_{u\in\mathcal{F}}\sum_{v\in\mathcal{F}}\boldsymbol{z}_{u,v} - \sum_{u\in\mathcal{F}}\sum_{v\in\mathcal{F}}\boldsymbol{z}_{u,v}\\ &= \sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{S}'_u}(\boldsymbol{x}_i+\boldsymbol{b}_i)\end{aligned}$$

In order to remove the random mask, the cloud server requests $r_{j,i}$ from each s_i. Then each worker s_i encrypts the secret share

$$D_{i,j} = \textbf{AE.encrypt}(sk_{i,S}, j||i||r_{j,i})$$

and sends the list $\{D_{i,j}\}_{j\in\mathcal{S}'_u}/\{i\}$ to S through f_u. Upon receiving encrypted secret shares from all surviving workers, S decrypts them and recover all r_i, then generates random masks to obtain the aggregation result $\sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{S}'_u}\boldsymbol{x}_i$.

C. FPTD Construction

In this part, we give detailed construction of FPTD. Note that the plaintext numbers are float while cryptographic primitives are performed on integers. We use a rounding factor of 10^6 to scale float numbers up to integers.

1) Initialization: Before collecting and transmitting data, each worker s_i initializes their weight as $w_i^0 = 1$ and distance as $d_i^0 = 0$. Then S initializes the public parameters and establishes symmetric keys via *One Time Initialization*.

2) Secure Truth Update: In this phase, FPTD securely aggregates weights and weighted truths to update the truths according to equation 1. In each time slot t, each s_i first collects sensory data $\boldsymbol{x}_i^t$ and uploads his weight w_i^{t-1} and weighted truth $w_i^{t-1}\times\boldsymbol{x}_i$ via ***SecAgg***. Notice that we can concatenate both values into one vector $\langle w_i^{t-1}\times\boldsymbol{x}_i, w_i^{t-1}\rangle$, resulting in only one call to ***SecAgg***.

3) Secure Weight Update: In this phase, S first returns truths $\boldsymbol{x}_t^*$ to all workers. After receiving truths, each s_i first updates his distance according to equation 2 to obtain d_i^t. Then each worker submits his distance to S via ***SecAgg***. After obtaining aggregated distance $\sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{S}'_u}d_i^t$, S computes $\log\left(\sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{S}'_u}d_i^t\right)$ and returns it to all workers. Upon receiving $\log\left(\sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{S}'_u}d_i^t\right)$, each worker updates his weight according to equation 3.

After initialization, FPTD executes Secure Truth Update phase and Secure Weight Update phase once in each time slot t.

VI. Security Analysis

Theorem 1. *Suppose there is at least one group and each group contains at least 3 workers providing different sensory data for each object. If all entities are semi-honest and there is no collusion between fog nodes and cloud server, the sensory*

data and weight will not be disclosed to any other entity under FPTD framework.

proof. In order to prove that FPTD is secure, we only need to prove the proposed ***SecAgg*** is secure. There are three types of attacks an adversary can potentially launch under the semi-honest assumption.

First, an adversary can compromise fog nodes. Once the adversary compromises some fog nodes $\mathcal{F}_c \subset \mathcal{F}$, he/she can obtain the combined view of these fog nodes which is a list $\{\sum_{i\in\mathcal{S}'_u} \boldsymbol{x}_i + \boldsymbol{b}_i\}_{u\in\mathcal{F}_c}$. However, since the random seeds used to recover the random masks are encrypted using the symmetric keys established between workers and the cloud server, the adversary is not able to recover any $\boldsymbol{x}_i$ or any partial result.

Second, an adversary can compromise the cloud server. Once the adversary compromises cloud server, he/she can remove all random masks and obtain both $\sum_{i\in\mathcal{S}'_u} \boldsymbol{x}_i + \sum_{v\in\mathcal{F}:u>v} \boldsymbol{z}_{u,v} - \sum_{v\in\mathcal{F}:u<v} \boldsymbol{z}_{u,v}$ for each f_u and $\sum_{u\in\mathcal{F}} \sum_{i\in\mathcal{S}'_u} \boldsymbol{x}_i$. However, since we assume no collusion between fog nodes and the cloud server, the cloud server cannot remove the zero-sum masks added by each fog node, therefore cloud server can only obtain aggregation result $\sum_{u\in\mathcal{F}} \sum_{i\in\mathcal{S}'_u} \boldsymbol{x}_i$.

Finally, collusion may happen between workers or fog nodes. Collusion in each group under the threshold t_u reveals nothing to corrupted workers since a t-out-of-n Shamir's secret sharing is information-theoretically secure against up to $t-1$ corrupted parties.

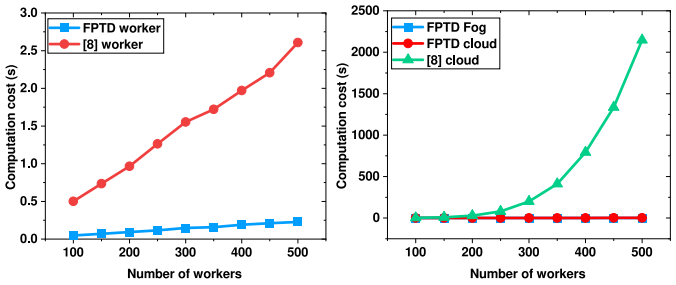
VII. Performance Analysis

Here we conduct the performance comparison between our scheme and [8] in terms of computation overhead and communication overhead. The simulation is built using a PC with 3.60Ghz Intel i7 and 16GB RAM running Windows 10. FPTD is implemented in Python. Specifically, we construct key agreement protocol from Elliptic-Curve over the NIST P-256 curve composed with SHA256, authenticated encryption from AES-GCM with a 128-bit key. In addition, we also implement [8] as a baseline scheme for comparison. Without loss of generality, we generate a dataset by sampling random numbers as ground truths and add different levels of Gaussian noise to simulate different levels of reliability of workers in MCS applications. Notice that, we regard a worker's reliability in different time slots as a dynamic variable since the ground truths for each object are rapidly changing over time.

For clarity, we present each entity's overhead in one time slot which consists of a *Secure Truth Update* and a *Secure Weight Update*. Besides we execute each experiment 50 times and present the average. Specifically, we assume the complete worker group is divided into groups equally. By default, we set the number of workers as 300, the number of fog nodes as 5, the number of objects as 100 and the drop rate as 0.05.

A. Computation

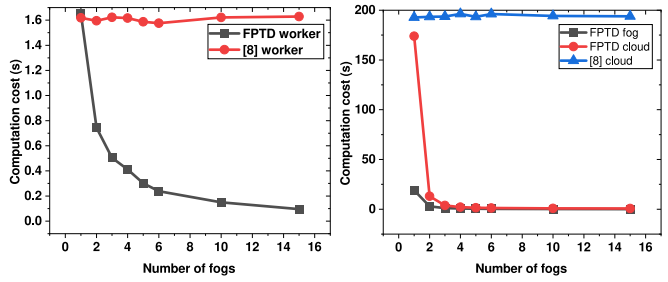
First, we consider varying number of workers which varies from 100 to 500 by the stride of 50. As shown in Fig.1, with the increasing number of workers, FPTD achieves better efficiency with respect to workers, fog and cloud. In particular, cloud computation overhead of [8] grows to be unacceptable when the number reaches 500 since the computation cost is beyond 30 minutes.



(a) Average time of workers (b) Average time of cloud and fog

Fig. 1. Average computation cost with varying workers

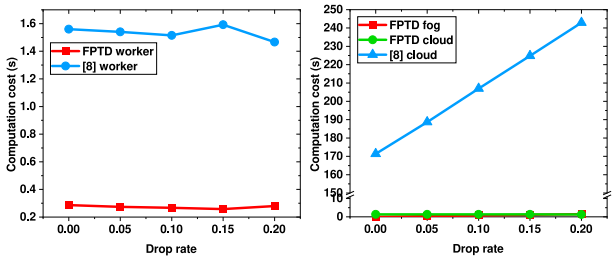
Next, we consider varying fog nodes which varies in range of $[1, 2, 3, 4, 5, 6, 10, 15]$. The result is presented in Fig.2. From Fig.2, we can observe that when the number of fog nodes is set to only, i.e., one group, FPTD achieves similar efficiency as [8] with respect to all entities. Further, FPTD's computation overhead drops significantly when the number of fog nodes grows. However, there is a turning point after which the computation overhead drops slowly with respect to all entities. The above observations show that it is enough for FPTD to adopt only a few fog nodes to gain significant efficiency improvement.



(a) Average time of workers (b) Average time of cloud and fog

Fig. 2. Average computation cost with varying fog nodes

Finally, we consider the effect of the drop rate. Specifically, we let the drop rate vary from 0.00 to 0.20 by the stride of 0.05. Here we consider only workers' dropouts before submitting their masked input since this is the worst case which brings most computation and communication overheads. As shown in Fig.3a, drop rate has little influence on worker's computation overhead since workers only need to upload shares of dropout workers. However, we can observe from Fig.3b that the computation overhead of cloud of [8] and fog of FPTD grows linearly since they need to recover the secret keys of dropout workers to cancel the zero-sum mask. Besides, the cloud of FPTD's computation overhead drops since the cloud needs to recover fewer random masks.

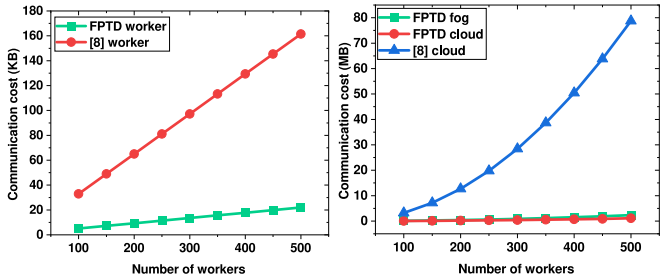


(a) Average time of workers (b) Average time of cloud and fog

Fig. 3. Average computation cost with varying drop rate

B. Communication

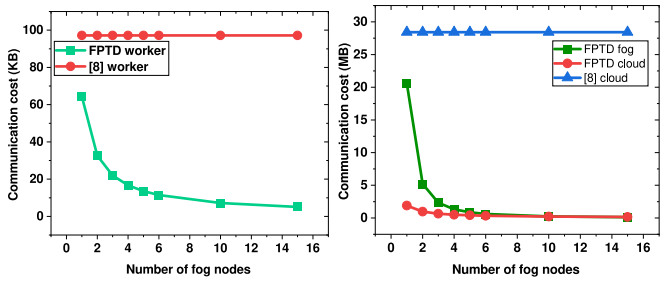
In this part, we give performance comparisons between our FPTD and [8] in terms of communication overhead. First, we present the performance with varying number of workers. As illustrated in Fig.4, workers' communication costs of both FPTD and [8] grow linearly to the number of workers. However, our FPTD grows significantly slower than [8] since the group size of FPTD is divided by the number of fogs.



(a) Average cost on each worker (b) Average cost on cloud and fog

Fig. 4. Average communication cost with varying number of workers

Next, we compare the performance with a varying number of fog nodes. We can observe from Fig.5a that the communication overhead of FPTD's workers drops when number of fog nodes grows. Moreover, the communication cost of FPTD's cloud is reduced since the cloud removes the burden of relaying the messages between workers. With more fog nodes, each fog node's communication overhead can also be significantly reduced since the group size is reduced.



(a) Average cost on each worker (b) Average cost on cloud and fog

Fig. 5. Average communication cost with varying number of fogs

Due to space limitations, we omit the comparison with varying drop rates and varying objects since these two factors don't influence the communication overhead heavily.

VIII. Conclusion

In this paper, we propose FPTD, a privacy-preserving truth discovery framework for real-time applications in mobile crowdsensing system. Our FPTD can utilize fog nodes to divide the complete group into many small groups, which gains significant efficiency improvement in terms of both computation and communication. Besides, we design a much more efficient and failure-robust secure aggregation protocol which can aggregate results from these small groups with no privacy compromise. However, we leave specific group strategies as future directions.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371 and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] X. Zhang, Z. Yang, and Y. Liu, "Vehicle-based bi-objective crowdsourcing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3420–3428, 2018.

[2] N. Lathia, V. Pejovic, K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "Smartphones for large-scale behavior change interventions," *IEEE Pervasive Computing*, vol. 12, no. 3, pp. 66–73, 2013.

[3] 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 *Proceedings of the 2014 International Conference on Management of Data (SIGMOD)*. ACM, 2014, pp. 1187–1198.

[4] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, "A confidence-aware approach for truth discovery on long-tail data," *Proceedings of the VLDB Endowment*, vol. 8, no. 4, pp. 425–436, 2014.

[5] 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 *Proceedings of the 2015 Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 183–196.

[6] 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*, 2019, DOI:10.1109/TDSC.2019.2919517.

[7] K. Xue, B. Zhu, Q. Yang, N. Gai, D. S. Wei, and N. Yu, "InPPTD: An lightweight incentive-based privacy-preserving truth discovery for crowd sensing systems," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4305–4316, 2020.

[8] Y. Liu, S. Tang, H.-T. Wu, and X. Zhang, "RTPT: A framework for real-time privacy-preserving truth discovery on crowdsensed data streams," *Computer Networks*, vol. 148, pp. 349–360, 2019.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.

[10] N. Gai, K. Xue, P. He, B. Zhu, J. Liu, and D. He, "An efficient data aggregation scheme with local differential privacy in smart grid," in *Proceedings of the 2020 International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2020, pp. 73–80.

[11] H. Jin, L. Su, and K. Nahrstedt, "Theseus: Incentivizing truth discovery in mobile crowd sensing systems," in *Proceedings of the 2017 International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2017, pp. 1–10.

[12] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[13] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, p. 612–613, 1979.