ChainSensing: A Novel Mobile Crowdsensing Framework with Blockchain

Xi Tao, *Member, IEEE* and Abdelhakim Senhaji Hafid, *Member, IEEE*

***Abstract*—Mobile crowdsensing (MCS) is a promising paradigm of large-scale sensing. A group of mobile users are recruited with their smart devices to accomplish various sensing tasks in specific areas. The mobility and intelligence of mobile users enable MCS to achieve a sufficient coverage ratio of sensing tasks or areas. Currently, MCS is generally proposed and implemented in a centralized way under a platform's control. However, this centralized structure is vulnerable to a single point of failure. The platform's failure leads to a shutdown of the entire system. In addition, there is a trust issue between the platform and mobile users because of computational transparency and financial security. It is possible that the platform manipulates the working process of MCS to obtain an improper gain. To overcome these problems, we propose a decentralized MCS framework named ChainSensing by leveraging blockchain. In ChainSensing, mobile users interact with blockchain via smart contracts to complete their operations, *e.g.*, publishing sensing tasks and submitting collected data. Since there are computationally intensive problems in ChainSensing, *e.g.*, path planning, path selection, and reward determination, it is significantly expensive to solve such problems in blockchain. Therefore, we propose to leverage smart devices and computing oracles to solve these problems. Specifically, we propose a heuristic algorithm to solve the path planning problem in smart devices of mobile users; we employ computing oracles to solve the path selection and reward determination problems. Finally, we conduct numerical simulations based on Ethereum to evaluate the performance of ChainSensing.**

***Index Terms*—Mobile crowdsensing, decentralized framework, blockchain, smart contract.**

I. Introduction

MOBILE crowdsensing (MCS), coined by Ganti *et al.* [1], is a promising paradigm of data collection in large-scale sensing [2]. A large number of mobile users are recruited with their smart devices (*e.g.*, smart phones, laptops, and wearables) to accomplish various sensing tasks. Compared with conventional methods of data collection (*e.g.*, wireless sensor networks), MCS is a cost-efficient method without investing in infrastructure and maintenance. In addition, the mobility of mobile users makes MCS suitable for different distributions of sensing tasks, *e.g.*, uniform and clustered distributions [3]. Due to these advantages, MCS has been widely used in many applications, *e.g.*, environment monitoring [4], transportation management [5], and healthcare [6], to name a few.

This research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) and Fonds de Recherche du Qubec - Nature et Technologies (FRQNT).

X. Tao and A. Hafid are with the Department of Computer Science and Operations Research, University of Montreal, Montreal, QC H3T 1J4, Canada (e-mails: xi.tao@umontreal.ca, ahafid@iro.umontreal.ca).

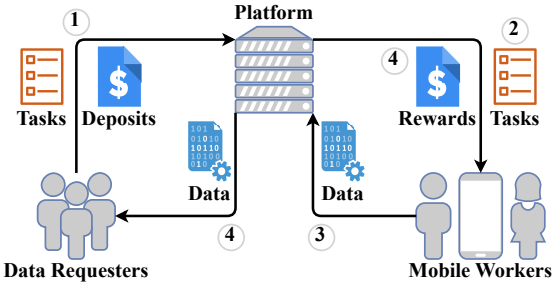


Fig. 1. A centralized structure of MCS.

It is a challenge in MCS to assign sensing tasks to mobile users, which is known as task allocation problem [7,8]. The assignment has a significant influence on the performance, *e.g.*, task coverage and data quality. In addition, an incentive mechanism is desired to attract mobile users to perform sensing tasks in budget-limited applications, which is known as incentive mechanism design [9,10]. Since mobile users have different devices and skills, it is important to evaluate the quality of collected data [11]. To address these problems, the existing solutions generally leverage a centralized platform to recruit mobile users, assign sensing tasks, evaluate data quality, and determine rewards [3,12,13]. Fig. 1 shows a centralized structure of MCS. Mobile users are classified as data requesters and mobile workers according to their roles in the sensing activity. The platform serves as an intermediary between data requesters and mobile workers. The platform first receives the information of sensing tasks and the corresponding deposits (rewards) from data requesters (step 1). Next, the platform assigns sensing tasks to mobile workers (step 2) and collects data from them (step 3). At last, the platform sends collected data to data requesters and distributes rewards to mobile workers (step 4). It is worth noting that all these processes (*e.g.*, flows of data and rewards) are under the platform's control.

Although the centralized structure enables the platform to employ global optimization algorithms to efficiently address these problems, there are several limitations in the centralized MCS. First, the platform may be a single point of failure in unexpected situations (*e.g.*, network attacks). The platform's breakdown leads the whole system to stop working and thus mobile users suffer losses. Second, there is a trust issue between the platform and mobile users from the perspectives of computational transparency and financial security. The lack of computational transparency means that mobile users do not know how the platform works. It is possible that the platform manipulates the working process to obtain an improper gain.

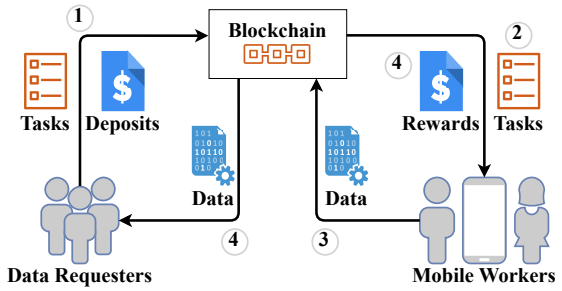


Fig. 2. A decentralized structure of MCS.

For instance, the platform can make a profit by receiving higher deposits from data requesters and distributing lower rewards to mobile workers. The financial security comes from the fact that mobile users store their deposits (rewards) in the platform.

To tackle these problems associated with the centralized structure, blockchain is employed to build the decentralized MCS [14,15]. Blockchain is introduced with Bitcoin (a digital cryptocurrency) as a decentralized digital ledger in a peer-to-peer (P2P) network [16]. This digital ledger records an increasing number of transactions into blocks and then appends the newly generated block to its previous block. These blocks are linked one by one as a chain; this is the origin of its name blockchain. Blockchain is maintained by independent nodes based on consensus protocols, *e.g.*, Proof-of-Work (PoW) and Proof-of-Stake (PoS). Since blockchain is decentralized, transparent, immutable, and secure, this technology is subsequently extended to a wider scope of applications, *e.g.*, Internet of things (IoTs) [17], smart cities [18], and edge computing [19]. Fig. 2 shows a decentralized structure of MCS based on blockchain.

Although blockchain is already applied in MCS, some key problems are not solved or clearly explained in existing studies. For instance, one important problem is where the computationally intensive problems of MCS (*e.g.*, task allocation) are solved (on-chain vs off-chain). To make the decentralized MCS more practical, we propose a novel decentralized MCS framework, named ChainSensing, based on blockchain with smart contracts. In ChainSensing, mobile users create their accounts (wallets) and interact with blockchain. Data requesters can publish their sensing tasks and deposit rewards to blockchain. Mobile workers who are willing to participate in the sensing activity make their sensing plans based on the task information. Since sensing tasks are generally location-dependent, mobile workers need to carefully design their travel paths as the sensing plans. We propose a heuristic algorithm, which runs in the smart devices of mobile workers, to plan travel paths. Next, mobile workers bid their sensing plans to blockchain. A smart contract deployed in blockchain makes a selection and determines rewards to the winners of the selection. However, it is significantly expensive to solve such computationally intensive problems in blockchain. For instance, the fundamental unit of computation in Ethereum is "gas" [20]. The more computation operations in a smart contract, the more gas is needed to execute the smart contract. In some cases, it is infeasible to execute a smart contract if it needs more gas than the maximum amount of gas set by Ethereum. To avoid the high cost in blockchain, we employ computing oracles to solve these problems. Finally, we evaluate our proposal via simulations on Ethereum.

The contributions of our paper can be summarized as follows:

- We propose a novel decentralized MCS framework (*i.e.*, ChainSensing), based on blockchain with smart contracts, to solve the problems associated with the centralized MCS (*e.g.*, a single point of failure and trust issue).
- We take into account the locations of sensing tasks and mobile workers in ChainSensing. We propose a heuristic algorithm for mobile workers to select sensing tasks and plan their travel paths.
- We employ computing oracles in ChainSensing to solve the computationally intensive problems to avoid the high cost in blockchain.
- We implement ChainSensing on Ethereum and conduct simulations to evaluate its performance.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes the structure and working process of our proposed framework. Section IV presents the solutions to our formulated problems. Section V evaluates our proposal via simulations. Finally, Section VI concludes the paper.

II. Related Work

In this section, we overview the related work with respect to centralized MCS and decentralized MCS.

A. Centralized MCS

In the centralized MCS, the platform has the information of mobile users and sensing tasks, thus the centralized MCS is able to meet different requirements with global optimization methods. For instance, the centralized MCS performs satisfactorily in terms of task coverage and energy efficiency [8,21].

The coverage of sensing tasks or areas is one important metric. It indicates the ability to successfully complete sensing tasks by mobile workers. Data requesters are willing to keep publishing their sensing tasks if the satisfactory coverage can be achieved. However, the definition of task coverage is not unique in different applications. Xiong *et al.* [22] proposed a framework named iCrowd, which considers the spatio-temporal coverage. To satisfy the requirement of task coverage, iCrowd makes a prediction of mobile workers' mobility based on their historical records and then recruits mobile workers according to the prediction. Zhang *et al.* [12] proposed a framework called CrowdRecruiter. The task coverage in CrowdRecruiter is defined as the number of covered cell towers in a time frame. Here, a cell tower is considered to be covered if there is at least one mobile worker making a phone call with it. By leveraging the prediction of phone calls, a small number of mobile workers can achieve a satisfactory task coverage.

In addition to task coverage, energy efficiency is another important metric. There are two widely used paradigms to reduce the energy consumption of MCS, *i.e.*, piggyback crowdsensing (PCS) and compressive crowdsensing (CCS). PCS reduces the

energy cost by exploiting the phone opportunities (*e.g.*, phone calls) [23]. For instance, a smart phone can upload data in its idle state in order to save the energy of screen lighting. EMC^3 [24] and EEMC [25] are two frameworks based on PCS. In EMC^3 and EEMC, the energy consumption of data transfer is reduced by leveraging the opportunities of phone calls. On the other hand, CCS is based on compressive sensing [26,27]. CCS first collects data from some specific areas and time slots. Next, it infers the data of unvisited areas to reconstruct a global distribution of data. CCS reduces the energy cost by collecting the minimum amount of data to meet the requirements of sensing tasks. Liu *et al.* [28] proposed a framework named UniTask, which is a CCS-based recovery scheme to improve the overall system utility.

In the existing centralized MCS frameworks, many fundamental problems (*e.g.*, task allocation) are efficiently solved. However, there are some challenges associated with the centralized structure. Since the centralized platform plays a role as the organizer, it can easily be the target of attackers. If the platform fails, the whole system crashes and stops working. This is known as a single point of failure. In addition, the trust issue between the centralized platform and mobile users is another challenge. Therefore, the decentralized MCS is desired to tackle these challenges.

B. Decentralized MCS

Blockchain is introduced with Bitcoin [16] as a decentralized digital ledger and then becomes a general technique to implement decentralized applications in various areas, *e.g.*, Internet of things (IoTs) [17], smart cities [18], and edge computing [19]. Ethereum [29] is a typical representative of blockchain platforms. There have been some studies on the decentralized MCS based on blockchain, *e.g.*, location privacy, incentive mechanism, and data quality.

Information privacy is significantly important in MCS because mobile users need to provide their private information (*e.g.*, location) during the sensing activity. Zou *et al.* [14] proposed a blockchain-based location-privacy-preserving framework named CrowdBLPS. There are two stages of CrowdBLPS including a preregistration stage and a final selection stage. Both two stages are deployed on blockchain via smart contracts. To protect the location privacy of mobile users, cloaked locations are used instead of exact locations. Jia *et al.* [30] proposed a blockchain-based framework with a coded method to protect location information (*e.g.*, longitude and latitude); the location information is coded to a fix-length digital number.

A variety of incentive mechanisms are well-designed in the centralized MCS [9]. Auction theory is widely used to design the incentive mechanism. However, it is a challenge to design the incentive mechanism in the decentralized MCS. Kadadha *et al.* [31] proposed a decentralized framework, named ABCrowd, based on blockchain and auction theory. In ABCrowd, an incentive mechanism, called Repeated-Single-Minded Bidder (R-SMB) auction, is designed to motivate mobile users to participate in the sensing activity. ABCrowd is implemented and deployed on a private Ethereum blockchain. Hu *et al.* [32] considered MCS as a sensory data market and proposed a fair incentive mechanism. In the proposed data market, mobile users are classified into monthly-pay users and instant-pay users. A three-stage Stackelberg game is designed to motivate the participation of mobile users. Finally, the incentive mechanism is deployed on blockchain.

Data quality is another concern of MCS. An *et al.* [33] proposed a blockchain-based model to assess data quality in the decentralized MCS. They used delegated proof of reputation (DPoR) as the consensus mechanism and deployed the model on blockchain via smart contracts. The simulation results show that the proposed model performs well in terms of data quality. Kadadha *et al.* [15] proposed a decentralized framework called SenseChain, which takes into account Quality of Information (QoI). In SenseChain, each mobile user has a specific QoI for each sensing task. Here, QoI is calculated based on reputation, time, and distance. The worker-task pair with the highest QoI has the largest probability to be selected as an assignment.

Although these are some existing studies on the decentralized MCS, some key problems are not solved or clearly explained. In this paper, we propose our solutions to three problems. The first problem is how mobile users make their sensing plans in the decentralized MCS. Since sensing tasks are location-dependent, mobile users have to travel around to perform them. In addition, the resources (*e.g.*, travel distance) of mobile users are limited. Therefore, mobile users should carefully design their travel paths as the sensing plans. In existing contributions, it is not clearly explained how the sensing plans of mobile users are determined. In this paper, we formulate the path planning problem and propose a heuristic algorithm to solve it. The second problem is related to the fact that it is expensive (sometimes infeasible) to solve computationally intensive problems in blockchain. However, the existing contributions (*e.g.*, [15,31]) implement all necessary computations in smart contracts. In this paper, we employ computing oracles to solve these problems to avoid the high cost in blockchain. Third, it is possible that the data recorded in blockchain is stolen by other mobile workers because blockchain is accessible to all mobile workers; this is known as the freeloading problem. The existing contributions do not consider this problem. In this paper, we propose that mobile workers first upload the hash value of their collected data and then they upload the real data after the reward determination is finished. The hash value can be used to verify the real data.

III. Framework Structure and Working Process

In this section, we describe in details the structure and working process of ChainSensing. In ChainSensing, we formulate the path planning, path selection, and reward determination problems. Table I shows the notations that are used in the rest of this paper.

A. Structure of ChainSensing

Fig. 3 shows the structure of ChainSensing. There are mobile users (data requesters or mobile workers), computing oracles, and blockchain (with smart contracts). In ChainSensing, blockchain is represented by a smart contract, which is

TABLE I
NOTATION DEFINITIONS.

Notations	Definitions
V	Set of sensing tasks
b_i	Reward of task v_i
t_i	Preset sensing time of task v_i
δ_i	Time window radius of task v_i
k_i	Required data number of task v_i
loc_i^t	Location of task v_i
W	Set of mobile workers
P_j	Travel path of worker w_j
h_j	Travel distance limit of worker w_j
l_j	Straight-line distance of worker w_j
f_j	Travel speed of worker w_j
$d(P_j)$	Travel distance of path P_j
τ_{ij}	Completion time of task v_i by worker w_j
x_j	Selection result of path P_j
q_{ij}	Data quality of task v_i from worker w_j
r_{ij}	Reward result of task v_i for worker w_j
u_j	Reward to worker w_j
loc_j^s	Start point of worker w_j
loc_j^e	Destination of worker w_j

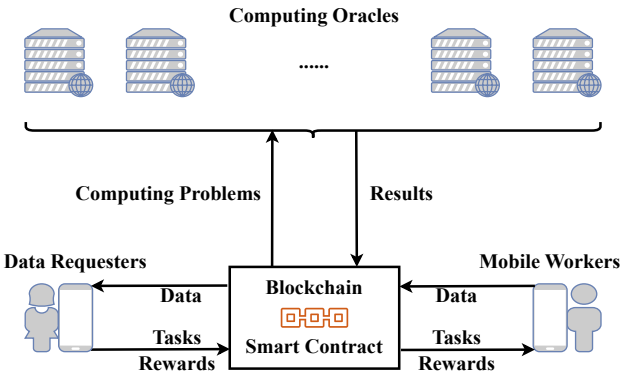


Fig. 3. Structure of ChainSensing.

the cornerstone of the system. Other components (*i.e.*, data requesters, mobile users, and computing oracles) interact with it to complete their operations. For instance, data requesters publish sensing tasks and deposit rewards to the smart contract. Mobile users access the task information via the smart contract and bid sensing plans to it. Since the smart contract has some computationally intensive problems to solve, *e.g.*, path selection problem, it employs a group of computing oracles to avoid high computational cost in blockchain. The smart contract leverages computing oracles to make a selection from the submitted sensing plans. An oracle is a trusted third-party entity that can be used to access data or perform computing tasks on behalf of smart contracts. There are two types of oracles: (a) data oracles: allow smart contracts to access data off-chain (*e.g.*, Ether exchange rate and temperature); and (b) computing oracles: perform computing tasks for smart contracts. Examples of oracles include Provable (formerly Oraclize; currently, it is the leading oracle) and Chainlink [34]. A computing oracle can be deployed in a data center (cloud) or an edge node [35].

B. Working Process of ChainSensing

Fig. 4 shows the working process of ChainSensing. There are different participants. A blockchain platform (*e.g.*, Ethereum) is leveraged to accommodate our smart contract. Since our smart contract is deployed in a specific platform, we directly use the consensus mechanisms in this platform. For instance, we use the PoW consensus (currently used by Ethereum) when we leverage Ethereum to deploy our smart contract. An organizer or initiator deploys the smart contract to the platform and controls the process of sensing activities. In ChainSensing, a complete cycle of sensing activities is divided into different phases and some operations are only allowed in certain phases. Mobile users are classified into two categories, *i.e.*, data requesters and mobile workers. In addition, computing oracles are used to solve computationally intensive problems. Next, we introduce the working process of ChainSensing phase by phase.

1) Deployment of Smart Contract: At the very beginning, the organizer deploys the smart contract in the blockchain platform (*e.g.*, Ethereum). Next, computing oracles register in the smart contract. It is worth noting that the organizer can define various ways of oracle registration in the smart contract. For instance, the registration is open to any computing oracle with certain computing power. To enhance security and trust, the organizer can provide a list of computing oracles and each oracle in the list is generally with high reputation. Thus, only computing oracles (trusted by the organizer) in the list can register in the smart contract. In addition, the smart contract can also create a reputation table to dynamically evaluate the computing oracles based on their performance in the sensing activities. The organizer can use different methods of registration according to the application. In this phase, data requesters and mobile workers create their own accounts (or wallets) that are used to interact with the smart contract. Each account contains the information of balance (*e.g.*, Ether in an Ethereum account). The identity of a mobile user (either data requester or mobile worker) can be confirmed by the account address (*i.e.*, hash of public key). After each phase is over, the organizer invokes a function in the smart contract to enter the next phase and the smart contract emits events to notify stakeholders.

2) Task Publishing: In this phase, data requesters publish their sensing tasks to the smart contract. The set of published tasks is represented by $V = \{v_1, v_2, ..., v_m\}$. For each task $v_i \in V$, an amount of deposit, denoted by b_i, is locked in the smart contract as the reward to mobile workers. Task v_i requires a sensing time that is denoted by t_i; it represents the expected time when task v_i is performed. To enable mobile workers to complete task v_i, there is a sensing interval δ_i. That is to say, the data collected within time window $[t_i - \delta_i, t_i + \delta_i]$ is acceptable to task v_i. In addition, task v_i may require multiple data samples; the number of samples is denoted by k_i. If any mobile worker is interested in the sensing activities, the task information can be accessed through the smart contract. The set of mobile workers is denoted by $W = \{w_1, w_2, ..., w_n\}$.

3) Path Planning: After mobile workers receive the task information, they need to make sensing plans by selecting sensing tasks. Since sensing tasks are generally location-dependent, mobile workers have to travel around to complete their plans with limited resources (*e.g.*, maximum travel dis-

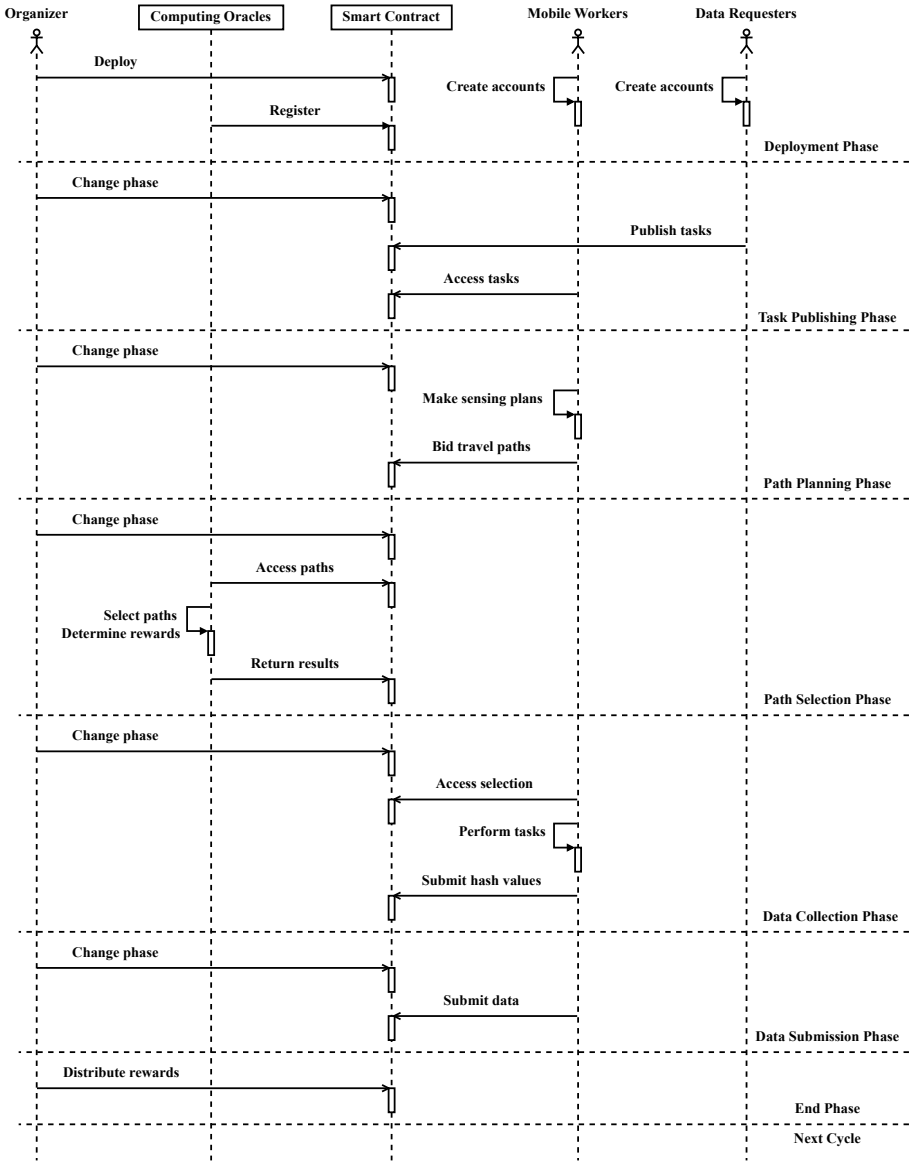


Fig. 4. Working process of ChainSensing.

tance); this is known as the path planning problem. For each mobile worker $w_j \in W$, the travel path is denoted by P_j. Fig. 5 shows the path planning problem of worker w_j. The travel path has a start point and a destination. We denote the straight-line distance from the start point to the destination by l_j. In addition, there is a maximum travel distance h_j associated with the travel path. To ensure that worker w_j is able to arrive at the destination, h_j should not be smaller than l_j. The travel speed of worker w_j is denoted by f_j. Since worker w_j desires to accomplish as many sensing tasks as possible and each selected task must be performed within its time window, the path planning problem of worker w_j is formulated as follows:

$$\text{max.} \sum_{v_i \in P_j} 1 \tag{1a}$$

$$\text{s.t.} \quad l_j \leq d(P_j) \leq h_j \tag{1b}$$

$$t_i - \delta_i \leq \tau_{ij} \leq t_i + \delta_i, \forall v_i \in P_j \tag{1c}$$

where $d(P_i)$ is the travel distance of path P_j and τ_{ij} is the time when task v_i is performed by worker w_j. $\sum_{v_i \in P_j} 1$ represents the number of completed tasks along path P_j. It is worth noting that it is not only one feasible strategy of mobile workers to perform as many tasks as possible when they plan their travel paths. They can design different strategies to select sensing tasks. Mobile workers solve the path planning problem by their devices (*e.g.*, smart phones and tablets) and then submit their sensing plans (travel paths) to the smart contract. They also submit deposits to the smart contract as penalty fees in case they cannot accomplish their assigned sensing tasks or they cannot upload the data on time.

4) Path Selection: In this phase, the smart contract needs to make a selection from the submitted sensing plans (travel paths); this is known as the path selection problem. After the path selection, the smart contract also needs to determine rewards to mobile workers. However, these computationally intensive problems are expensive to be solved directly in the

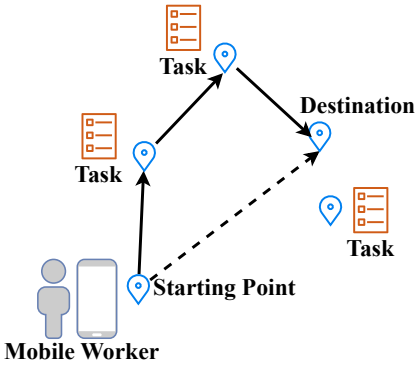


Fig. 5. Path planning problem.

smart contract. Therefore, we leverage computing oracles to solve them. Since there are multiple oracles registered in the smart contract, we can design various methods to employ one or more oracles. For instance, we can randomly select one oracle. The computing oracle receives the information of these problems and then solve them. Next, it returns the results to the smart contract. To ensure that the smart contract can receive the results, we may impose some penalties on the computing oracle if the results are not returned on time.

In the path selection problem, the objective is to complete all sensing tasks with a minimum number of selected sensing plans (travel paths). Thus, the path selection problem is formulated as follows:

$$\text{min.} \quad \sum_{w_j \in W} x_j \tag{2a}$$

$$\text{s.t.} \quad \sum_{P_j : v_i \in P_j} x_j \geq k_i, \forall v_i \in V \tag{2b}$$

$$x_j \in \{0, 1\}, \forall w_j \in W \tag{2c}$$

where $x_j = 1$ indicates the travel path of worker w_j, *i.e.*, path P_j, is selected; otherwise, it is not selected. Hence, $\sum_{w_j \in W} x_j$ is the number of selected travel paths. Constraint (2b) indicates that task v_i requires at least k_i data samples. It may receive more than k_i data samples from mobile workers. However, only k_i mobile workers can receive their rewards for performing task v_i because the deposit from the data requester of task v_i is only enough to issue these k_i rewards.

We propose to select the subset with k_i elements based on data quality. According to [36], data quality can be identified in four dimensions: accuracy, completeness, consistency, and timeliness. In this paper, we consider data quality from the perspective of timeliness. Task v_i is expected to be performed within a time window that is centered at time t_i. If worker w_j performs it at time τ_{ij}, the data quality is evaluated by

$$q_{ij} = 1 - \frac{|\tau_{ij} - t_i|}{\delta_i} \tag{3}$$

where data quality q_{ij} is within $[0, 1]$. When time τ_{ij} is closer to time t_i (*i.e.*, better performance in timeliness), data quality q_{ij} is closer to 1; otherwise, data quality q_{ij} is closer to 0. It is worth noting that we can use different methods to evaluate data quality by modifying (3) accordingly.

Next, we sorts mobile workers, who compete for task v_i, by the data quality from largest to smallest. Only k_i mobile workers are selected and awarded by performing task v_i. We use r_{ij} to represent the selection result of worker w_j for task v_i. $r_{ij} = 1$ indicates that worker w_j is selected and awarded for performing task v_i; otherwise, r_{ij} equals 0. The total reward to worker w_j, denoted by u_j, is determined by

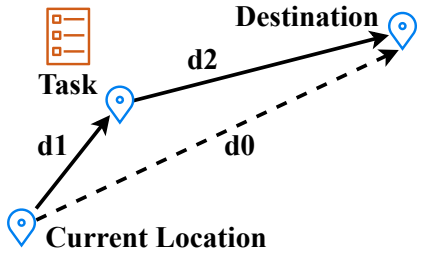


Fig. 6. An example of calculating the additional distance.

$$u_j = \sum_{v_i \in P_j} (r_{ij} \cdot b_i). \tag{4}$$

At the end of this phase, the computing oracle returns the results to the smart contract.

5) Data Collection: In this phase, mobile workers access the selection result from the smart contract. The selected workers start to move around to perform the sensing tasks along their travel paths and collect data at the locations of these tasks. After the data collection, mobile workers submit hash values of their collected data. Since the information recorded in the smart contract is accessible to all mobile workers, it is possible that the uploaded data is stolen by other mobile workers; this is known as the freeloading problem. To solve this problem, mobile workers first upload the hash values of their collected data.

6) Data Submission: In this phase, mobile workers submit their real data to the smart contract and then the smart contract checks the real data of each mobile worker with the corresponding hash value. If the check fails, this mobile worker receives no reward and loses his own deposit.

7) Reward Distribution: In this phase, the smart contract completes transactions if there is no problem in all the previous phases. The rewards are distributed to the wallets of mobile workers and data requesters can access the revealed data. The sensing activities enter the next cycle.

IV. Proposed Solutions

In this section, we propose solutions to the path planning, path selection, and reward determination problems. In addition, we analyze the complexity of path planning and path selection problems. At last, we give the details of the smart contract that implements ChainSensing.

A. Path Planning Problem

We briefly prove that path planning problem in (1) is NP-hard.

Proof. We reduce a known NP-hard problem, *i.e.*, orienteering problem [37], to an instance of our formulated problem in (1). In the orienteering problem, a player needs to plan a tour to visit a number of tasks that are associated with certain game

Algorithm 1: Solution to path planning problem.

Input: V, $\{loc_i^t|\forall v_i \in V\}$, $\{t_i|\forall v_i \in V\}$, $\{\delta_i|\forall v_i \in V\}$, w_j, loc_j^s, loc_j^e, h_j, l_j, f_j
Output: P_j

1 $P_j \leftarrow \emptyset$
2 $location \leftarrow loc_j^s$
3 $distance \leftarrow 0$
4 **while** $distance \leq h_j - l_j$ **do**
5 Get task set of all feasible tasks
6 **if** *there is no feasible task* **then**
7 **break**
8 **else**
9 Select task v_i with the smallest cost $\hat{d}(location, loc_i^t) \cdot |\tau_{ij} - t_i|$
10 $location \leftarrow loc_i^t$
11 $distance \leftarrow distance + \hat{d}(location, loc_i^t)$
12 Add v_i into P_j
13 **return** P_j

points. The objective is to maximize the final point with a limited travel distance. The orienteering problem is already proved to be NP-hard [37]. Let us construct an instance of the path planning problem to solve the orienteering problem. We assume that each sensing task has a game point of 1. The time window of each sensing task is open during the sensing activities. Hence, this constructed instance of the path planning problem is exactly an orienteering problem. We conclude that the path planning problem in (1) is also NP-hard. □

Because the path planning problem is NP-hard, the search space to find the optimal solution is huge. In addition, the path planning problem is independently solved by the device of each mobile worker with limited computational resources. Therefore, an efficient algorithm is desired to solve the path planning problem formulated in (1); we propose a heuristic algorithm (see Alg. 1).

In Alg. 1, lines 1-3 initialize the travel path, current location, and additional travel distance, respectively. To perform a sensing task, worker w_j has to travel an additional travel distance. Fig. 6 shows an example of calculating the additional travel distance. In this example, the additional travel distance is computed to be $(d_1 + d_2 - d_0)$. Since worker w_j has a travel distance limit h_j and the straight-line distance from its start point to its destination is l_j, the maximum additional travel distance is limited by $(h_j - l_j)$. If the current additional travel distance does not exceed this limit (line 4), the set of feasible sensing tasks is built (line 5). Here, a feasible sensing task indicates it can be performed within its time window and worker w_j can perform it within the additional travel distance limit. If there is no feasible sensing task (line 7), *i.e.*, this set is empty, the while loop ends (line 8); otherwise, the sensing task with the smallest cost is selected (line 9). We consider the cost from two perspectives, *i.e.*, distance and timeliness. $\hat{d}(location, loc_i^t)$ is the additional travel distance of worker w_j to perform task v_i. $|\tau_{ij} - t_i|$ is used to measure the corresponding timeliness. To reduce the cost, both the additional travel distance and timeliness should be minimized. If task v_i is performed by worker w_j, the current location, additional travel distance, and travel path of worker w_j are updated (lines 10-12). At last, line 13 returns the travel path of worker w_j, *i.e.*, P_j.

Algorithm 2: Solution to path selection problem.

Input: V, $\{k_i|\forall v_i \in V\}$, W, $\{P_j|\forall w_j \in W\}$
Output: $\{x_j|\forall w_j \in W\}$

1 **for** j *from* 1 *to* n **do**
2 $x_j \leftarrow 0$
3 Set up U
4 $C \leftarrow \emptyset$
5 **while** $U \nsubseteq C$ **do**
6 Select path P_j with the largest number of desired data samples in $(U - C)$
7 $C \leftarrow C \cup P_j$
8 $x_j \leftarrow 1$
9 **return** $\{x_j|\forall w_j \in W\}$

In the worst case, all m sensing tasks are added into the travel path and thus there are m iterations (lines 4-12). In each iteration, we consider all unvisited sensing tasks to find the sensing task with the smallest cost (line 9). At the beginning, there are m unvisited tasks. Next, there is one task added to the travel path at each iteration. Therefore, the time complexity of Alg. 1 is $O(m^2)$ (*i.e.*, $m + ... + 2 + 1$).

B. Path Selection Problem

We briefly prove that path selection problem in (2) is NP-hard.

Proof. If the number of desired data samples for each task is 1, the path selection problem in (2) is an instance of the set cover problem. It is known that the optimization version of set cover problem is NP-hard. Therefore, the path selection problem is also NP-hard as a generalized set cover problem. □

We propose a greedy-based algorithm to solve the path selection problem (see Alg. 2). Lines 1-2 initialize the selection status of travel paths. Line 3 sets up the target union. Here, the target union is a set of all desired data samples. For instance, if tasks v_1, v_2, and v_3 desire 1, 2, and 3 data samples, respectively. The corresponding target union is $\{v_1, v_2, v_2, v_3, v_3, v_3\}$. Line 4 initializes the current union of data samples. If the current union does not cover the target union (line 5), we select the travel path with the largest number of desired data samples in the union of unvisited tasks, denoted by $(U - C)$, in line 6. Next, we add the data samples along the selected travel path into the current union (line 7) and set its selection status to be 1 (line 8). Finally, we return the result of selection when the target union is fully covered (line 9).

The initialization of selection status in lines 1-2 is $O(n)$, when all n mobile workers submit their travel paths. In the worst case, we have to select all n travel paths to cover the target union and thus there are n iterations (lines 5-8). At each iteration, we check every unselected travel path to find the travel path with the largest number of desired data samples

Algorithm 3: Solution to reward determination problem.

Input: V, $\{b_i|\forall v_i \in V\}$, $\{k_i|\forall v_i \in V\}$, W, $\{P_j|\forall w_j \in W\}$, $\{x_j|\forall w_j \in W\}$
Output: $\{u_j|\forall w_j \in W\}$

1 **for** j *from* 1 *to* n **do**
2 $u_j \leftarrow 0$
3 **for** i *from* 1 *to* m **do**
4 Get set of competitors for task v_i as C_i
5 Sort C_i in an increasing order of data quality
6 **for** j *from* 1 *to* n **do**
7 **if** x_j *equals* 1 *and task* v_i *is in Path* P_j *and worker* w_j *wins in* C_i **then**
8 $u_j \leftarrow u_j + b_i$
9 **return** $\{u_j|\forall w_j \in W\}$

in $(U - C)$ in line 6. Therefore, the overall time complexity of Alg. 2 is $O(n^2)$ (*i.e.*, $n + ... + 2 + 1$).

C. Reward Determination Problem

The solution to the reward determination problem is simple and straightforward. For each sensing task $v_i \in V$, we first list all mobile workers who include task v_i in their travel paths; these mobile workers are the competitors for task v_i. Next, we select winners one by one from the competitors according to data quality. Alg. 3 shows the details of our proposed solution. Lines 1-2 initialize the rewards to mobile workers. For each sensing task $v_i \in V$ (line 3), we obtain the set of its competitors (line 4) and sort them in an increasing order with respect to data quality (line 5). To find the winners of task v_i, each mobile worker $w_j \in W$ is evaluated (line 6). If worker w_j meets certain conditions (line 7), the reward of task v_i, *i.e.*, b_i, is distributed to worker w_j (line 8). There are three conditions. First, the travel path of worker w_j is selected (*i.e.*, $x_j = 1$). Second, task v_i is included in the travel path of worker w_j. Third, worker w_j is a winning competitor of task v_i. Finally, line 9 returns the result of final rewards to mobile workers.

The initialization of rewards to mobile workers in lines 1-2 is $O(n)$, when all n mobile workers are considered. Since we have m sensing tasks, there are m iterations (lines 3-8). At each iteration, we check all n mobile worker to set up the set of competitors (line 4). The time complexity of sorting this set by sorting algorithms (*e.g.*, merge sort) is $O(n \log n)$ in line 5. In the worst case, we traverse all n mobile workers to find the k_i winners of task v_i (lines 6-8); this step is $O(n)$. Therefore, the overall time complexity of Alg. 3 is $O(mn \log n)$ taking into account all m iterations.

D. Smart Contract

Fig. 4 shows the working process of our proposed framework. For more details of interactions and operations in our smart contract, Table II shows its implementation details. The smart contract is implemented by using Solidity. There are different types of components in the smart contract, *e.g.*, variable, event, modifier, and function. Variables have their own data types. There are many built-in data types, *e.g.*, uint (unsigned integer), address, and mapping. On the other hand, we can define new data types. For instance, struct is one user-defined type to represent a collection of variables under a single name and enum is one way to create a user-defined type that restricts a variable to have one of a few predefined values. Events allow the convenient usage of logging. When events are emitted, the logs are stored in blockchain and they are accessible. Thus, we can use events to inform mobile workers of the current phase. Modifiers are special functions that are used to modify the behaviour of a function. For instance, we can add a prerequisite to a function via modifiers. Functions are reusable code that can be called to complete some tasks.

TABLE II
IMPLEMENTATION DETAILS OF SMART CONTRACT.

Notations	Types
Task {longitude, latitude, reward, time, interval, number}	variable(struct)
Path {tasks, deposit, reward, selection, hash, data}	variable(struct)
organizer	variable(address)
tasks	variable(mapping)
paths	variable(mapping)
oracles	variable(mapping)
Phase {Deployment, Publishing, Bidding, Selection, Collection, Submission, Done}	variable(enum)
currentPhase	variable(Phase)
RegistrationStarted()	event
PublishingStarted()	event
BiddingStarted()	event
SelectionStarted()	event
CollectionStarted()	event
SubmissionStarted()	event
Ended()	event
validPhase()	modifier
onlyOrganizer()	modifier
constructor()	function
changePhase()	function
registerOracles()	function
publishTasks()	function
bidPaths()	function
returnSelection()	function
submitHash()	function
submitData()	function
distributeRewards()	function

In Table II, "Task" is an user-defined data type to store variables related to a sensing task, *e.g.*, location (longitude and latitude). "Path" is a data type defined for the travel path of one mobile worker. "organizer" represents the organizer address. "tasks", "paths", and "oracles" store the information of sensing tasks, travel paths, and computing oracles, respectively. "Phase" is an user-defined data type to enumerate all phases and "currentPhase" is used to track the current phase. Since we emit an event when the sensing activities enter a new phase, we define seven events corresponding to seven phases, *e.g.*, "RegistrationStarted()". To add prerequisites to some functions, we define two modifiers. "validPhase()" is used to check whether a function is feasible in the current phase and "onlyOrganizer()" indicates that a function can only be executed by the organizer. "constructor()" is the constructor function to perform the initialization when the smart contract is deployed. "changePhase()" is used by the organizer to change the current phase to the next phase. "registerOracles()" is a function that allows computing oracles to register in the smart

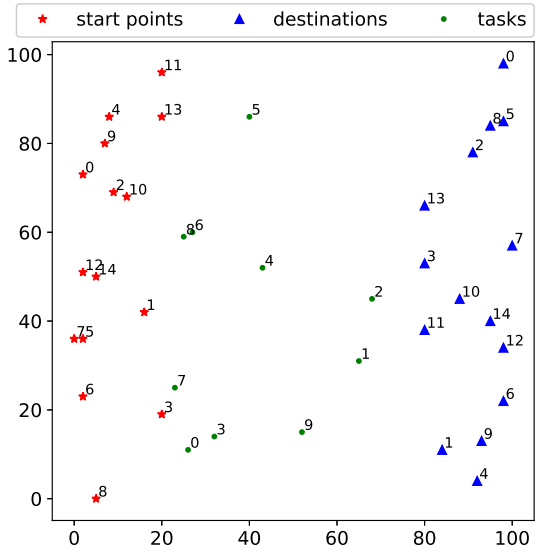


Fig. 7. Locations of sensing tasks and mobile workers.

contract. "publishTasks()" is used by data requesters to publish their sensing tasks. "bidPaths()" allows mobile workers to bid their sensing plans (travel paths). "returnSelection()" is used by computing oracles to return the results of path selection to the smart contract. Mobile workers use two functions to submit the hash of data (with "submitHash()") and the real data (with "submitData()"). Finally, a function ("distributeRewards()") is automatically invoked to transfer the total benefits of mobile workers (sensing rewards and their own deposits) to their wallets when the smart contract enters the last phase.

V. Numerical Results and Discussion

In this section, we evaluate our proposal via simulations. More specifically, we show the results of path planning, path selection, and reward determination. In addition, we present the cost of operations in ChainSensing.

A. Simulation Settings

We set up the sensing region as a square area with side length of 100 meters. 10 sensing tasks and 15 mobile workers are distributed in this area. Mobile workers start from the left side and move to their destinations that are located at the right side. Sensing tasks are located in the middle area. Fig. 7 shows the locations of sensing tasks and mobile workers.

The travel speed of each mobile worker is 1 meter per second (*i.e.*, normal walking speed of people) and the travel distance limit of each mobile worker is 400 meters. To measure the travel distance of mobile workers, Euclidean distance is used. The sensing time of each sensing task is randomly selected from $[100, 150]$ (unit is second) and the interval of each sensing task is 100 seconds. The reward of each sensing task is 1. The required number of data samples for each sensing task is 3. Finally, the settings of all simulation parameters can be found in Table III. To evaluate our proposal, we deploy our smart contract on Ethereum via Remix. The transaction cost of the deployment is 1934270 gas.

TABLE III
Simulation Parameters.

Parameter	Value
Sensing area	100 m ×100 m
Number of sensing tasks	10
Number of mobile workers	15
Travel speed of each worker	1 m/s
Travel distance limit of each worker	400 m
Sensing time of each task	100-150 s
Sensing interval of each task	100 s
Reward of each task	1
Number of desired data samples for each task	3
Blockchain platform	Ethereum

TABLE IV
Result of Bidding Paths.

Worker	Path	Transaction Cost (gas)
0	5, 2, 9, 1, 4, 6, 8	435231
1	9, 2, 1, 3, 0, 7, 4, 6, 8	462915
2	5, 2, 9, 1, 4, 6, 8	420231
3	2, 5, 4, 1, 9, 3	379657
4	4, 1, 2, 9, 3, 0, 7, 6, 8	462915
5	4, 2, 3, 1, 9	339083
6	7, 1, 6, 4, 2, 9, 3	420231
7	2, 6, 8, 4, 1, 9, 3	420231
8	2, 3, 1, 4, 6	339083
9	4, 1, 2, 9, 3, 0, 7, 6, 8	462915
10	6, 2, 7, 1, 4, 8	379657
11	2, 3, 1, 4, 6, 8	379657
12	2, 6, 4, 1, 9, 3	379657
13	5, 0, 3, 1, 2, 4, 6, 8	422341
14	2, 8, 6, 4, 1, 9, 3	420231

B. Result of Publishing Tasks

In the task publishing phase, data requesters publish their sensing tasks to the smart contract. Since the data structure of each sensing task is predetermined (*e.g.*, location, sensing time, and reward), the transaction cost to publish a sensing task is fixed at 145007 gas.

C. Result of Bidding Paths

In the path planning phase, mobile workers plan their travel paths with the smart devices via Alg. 1 and then bid their travel paths to the smart contract. Since travel paths contain different sensing tasks, the transaction cost to bid a travel path is also different; the results are shown in Table IV.

The transaction cost increases as the number of sensing tasks grows because there are more data transferred to the smart contract. For instance, the travel path of mobile worker #1 has the largest number of 9 sensing tasks and the corresponding transaction cost also has the largest value of 462915 gas. On the contrary, there are only 5 sensing tasks in the travel path of mobile worker #5 (the smallest number) and the transaction cost of bidding his path has the smallest value of 339083 gas. When the travel paths include the same numbers of sensing tasks, the transaction costs are the same even if the sensing tasks are not exactly the same. For example, the travel paths of mobile workers #6 and #7 have different task sets: (#7, #1, #6, #4, #2, #9, #3) and (#2, #6, #8, #4, #1, #9, #3). However, they have the same transaction cost of 420231 gas because they have the same number of sensing tasks (*i.e.*, 7).

TABLE V
RESULT OF WINNERS.

Task	Competitor	Winner
0	1, 4, 9, 13	13, 1, 9
1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	2, 13, 3
2	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	4, 9, 2
3	1, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14	1, 13, 9
4	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	3, 2, 13
5	0, 2, 3, 13	3, 2, 13
6	0, 1, 2, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14	2, 13, 1
7	1, 4, 6, 9, 10	1, 9, 4
8	0, 1, 2, 4, 7, 9, 10, 11, 13, 14	2, 13, 1
9	0, 1, 2, 3, 4, 5, 6, 7, 9, 12, 14	9, 4, 2

TABLE VI
RESULT OF REWARDS TO WORKERS.

Worker	Selection	Reward
0	No	0
1	Yes	5
2	Yes	7
3	Yes	3
4	Yes	3
5	No	0
6	No	0
7	No	0
8	No	0
9	Yes	5
10	No	0
11	No	0
12	No	0
13	Yes	7
14	No	0

D. Result of Selecting Paths

In ChainSensing, we propose to leverage computing oracles to solve the path selection and reward determination problems. In our simulation, we use our own computer as the trusted computing oracle. In real-life setup, commercial oracles can be used (*e.g.*, Provable and Chainlink) [34]. After the calculation, we submit the results of path selection and reward determination to the smart contract. The submission has a transaction cost of 424543 gas.

Table V shows the sets of competitors and winners for each sensing task. The competition for some sensing tasks is so fierce (*e.g.*, tasks #1, #2, and #4) that all mobile workers are fighting for them. However, there are some sensing tasks with a small number of competitors (*e.g.*, tasks #0 and #5). Since each sensing task requires 3 data samples, there are 3 final winners of each sensing task.

Table VI shows the results of path selection and rewards to mobile workers. There are 6 mobile workers that are selected with their travel paths, *i.e.*, workers (#1, #2, #3, #4, #9, #13). Each selected mobile worker receives a reward. The final reward of one mobile worker is determined by the sensing tasks that he wins (see Table V). For instance, the travel path of mobile worker #1 contains 9 sensing tasks (#9, #2, #1, #3, #0, #7, #4, #6, #8). However, mobile worker #1 only wins 5 of them, *i.e.*, (#3, #0, #7, #6, #8), according to Table V. Therefore, the final reward to mobile worker #1 is 5.

To show the details of how these winning mobile workers perform the sensing tasks along their travel paths, Fig. 8 presents the travel paths of winners (*i.e.*, workers #1, #2, #3, #4, #9, and #13). Mobile workers #4 and #9 have similar travel paths, start locations, and destinations. That is to say, they are competitors with each other. However, both travel paths are selected as winners because they can finish 9 sensing tasks. Although there is a competition among mobile workers, the one who can accomplish more sensing tasks always has an advantage over others.

TABLE VII
RESULT OF DATA SUBMISSION.

Winner	Data	Transaction Cost (gas)
1	9, 2, 1, 3, 0, 7, 4, 6, 8	37131
2	5, 2, 9, 1, 4, 6, 8	34587
3	2, 5, 4, 1, 9, 3	33283
4	4, 1, 2, 9, 3, 0, 7, 6, 8	37131
9	4, 1, 2, 9, 3, 0, 7, 6, 8	37131
13	5, 0, 3, 1, 2, 4, 6, 8	35827

E. Result of Submitting Data

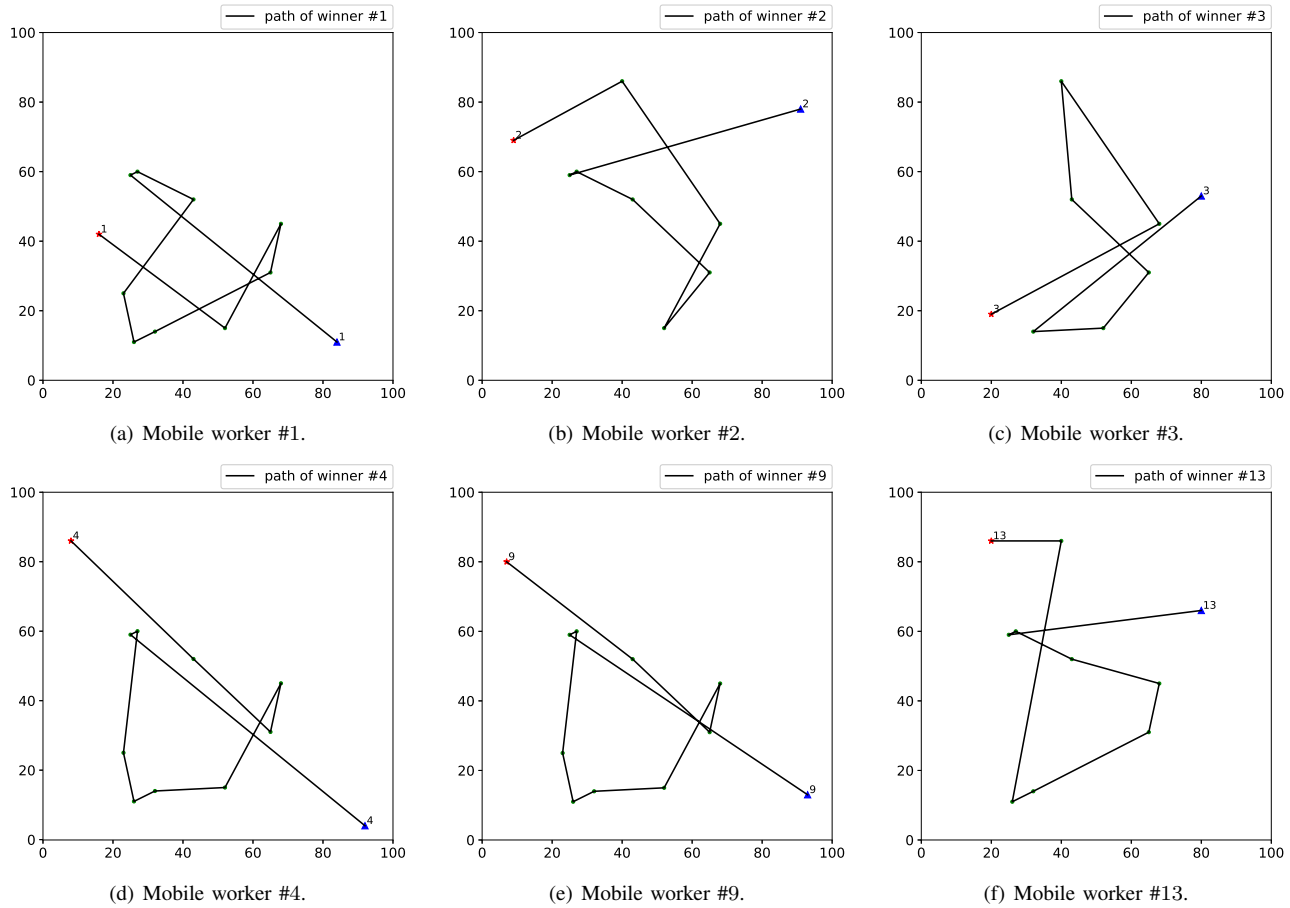
To prevent the freeloading problem, selected mobile workers first submit the hash value of collected data in the data collection phase. Mobile workers compute the hash by applying hash functions (*e.g.*, Keccak-256) to the collected data. Since the hash value has a fixed length, the transaction cost to submit the hash value by different mobile workers is the same; it is 23654 gas.

In the data submission phase, mobile workers start to submit the real data along their travel paths. The real data of each sensing task in our simulations is just an integer number (the index of each sensing task). It is worth noting that we can use different data types in other applications. Table VII shows the transaction cost to submit the real data. We observe that the transaction cost increases with the amount of submitted data.

In the last phase, the smart contract checks the submitted data with the corresponding hash values. If the check passes, the smart contract distributes rewards to selected mobile workers.

VI. CONCLUSION

In this paper, we investigated the problems of centralized MCS, *e.g.*, single point of failure and trust issue. To solve these problems, we proposed a novel decentralized MCS framework, named ChainSensing, based on blockchain with smart contracts. We defined the structure and working process of ChainSensing. The interactions between mobile users and ChainSensing is illustrated in details. In ChainSensing, we have to solve several computationally intensive problems including path planning, path selection, and reward determination problems. We proposed the corresponding solutions to these problems. More specifically, we proposed a heuristic solution running on mobile users' devices to the path planning problem. We leveraged computing oracles to solve the path selection and reward determination problems to avoid the high cost in blockchain. Greedy-based solutions are proposed to solve these two problems in computing oracles. Finally, we evaluated our proposal via simulations on Ethereum. The



(a) Mobile worker #1. (b) Mobile worker #2. (c) Mobile worker #3.

(d) Mobile worker #4. (e) Mobile worker #9. (f) Mobile worker #13.

Fig. 8. Travel paths of winners.

results show that our proposed framework is able to publish sensing tasks on behalf of data requesters and motivate mobile workers by rewards to perform sensing tasks.

REFERENCES

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[2] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[3] X. Tao and W. Song, "Location-dependent task allocation for mobile crowdsensing with clustering effect," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1029–1045, 2019.

[4] Z. Pan, H. Yu, C. Miao, and C. Leung, "Crowdsensing air quality with camera-enabled mobile devices," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 4728–4733.

[5] X. Wang, Z. Ning, X. Hu, E. C. H. Ngai, L. Wang, B. Hu, and R. Y. Kwok, "A city-wide real-time traffic management system: Enabling crowdsensing in social Internet of vehicles," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 19–25, 2018.

[6] S. Jovanovic, M. Jovanovic, T. Skoric, S. Jokic, B. Milovanovic, K. Katzis, and D. Bajic, "A mobile crowd sensing application for hypertensive patients," *Sensors*, vol. 19, no. 2, p. 400, 2019.

[7] B. Guo, Y. Liu, L. Wang, V. O. K. Li, J. C. K. Lam, and Z. Yu, "Task allocation in spatial crowdsourcing: Current state and future directions," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1749–1764, 2018.

[8] W. Guo, W. Zhu, Z. Yu, J. Wang, and B. Guo, "A survey of task allocation: Contrastive perspectives from wireless sensor networks and mobile crowdsensing," *IEEE Access*, vol. 7, pp. 78 406–78 420, 2019.

[9] H. Gao, C. H. Liu, W. Wang, J. R. Zhao, Z. Song, X. Su, J. Crowcroft, and K. K. Leung, "A survey of incentive mechanisms for participatory sensing," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 918–943, 2015.

[10] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij, "A survey of incentive techniques for mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 370–380, 2015.

[11] F. Restuccia, N. Ghosh, S. Bhattacharjee, S. K. Das, and T. Melodia, "Quality of information in mobile crowdsensing: Survey and research challenges," *ACM Transactions on Sensor Networks*, vol. 13, no. 4, pp. 34:1–34:43, 2017.

[12] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014, pp. 703–714.

[13] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "ActiveCrowd: A framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, 2017.

[14] S. Zou, J. Xi, H. Wang, and G. Xu, "CrowdBLPS: A blockchain-based location-privacy-preserving mobile crowdsensing system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4206–4218, 2020.

[15] M. Kadadha, H. Otrok, R. Mizouni, S. Singh, and A. Ouali, "SenseChain: A blockchain-based crowdsensing framework for multiple requesters and multiple workers," *Future Generation Computer Systems*, vol. 105, pp. 650–664, 2020.

[16] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008 (accessed June 1, 2020). [Online]. Available: https://bitcoin.org/bitcoin.pdf

[17] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2019.

[18] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, "A survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2794–2830, 2019.

[19] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019.

[20] A. M. Antonopoulos and G. Wood, *Mastering ethereum: Building smart contracts and dapps*. O'Reilly Media, 2018.

[21] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: State-of-the-art and future opportunities," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3747–3757, 2018.

[22] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "iCrowd: Near-optimal task allocation for piggyback crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[23] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, "Piggyback CrowdSensing (PCS): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, 2013, pp. 7:1–7:14.

[24] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, "EMC^3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1355–1368, 2015.

[25] H. Xiong, D. Zhang, L. Wang, J. P. Gibson, and J. Zhu, "EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, pp. 39:1–39:26, 2015.

[26] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[27] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[28] Z. Liu, Z. Li, and K. Wu, "UniTask: A unified task assignment design for mobile crowdsourcing-based urban sensing," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6629–6641, 2019.

[29] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[30] B. Jia, T. Zhou, W. Li, Z. Liu, and J. Zhang, "A blockchain-based location privacy protection incentive mechanism in crowd sensing networks," *Sensors*, vol. 18, no. 11, pp. 3894:1–3894:13, 2018.

[31] M. Kadadha, R. Mizouni, S. Singh, H. Otrok, and A. Ouali, "*ABCrowd*: An *A*uction mechanism on *B*lockchain for spatial *Crowd*sourcing," *IEEE Access*, vol. 8, pp. 12 745–12 757, 2020.

[32] J. Hu, K. Yang, K. Wang, and K. Zhang, "A blockchain-based reward mechanism for mobile crowdsensing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 178–191, 2020.

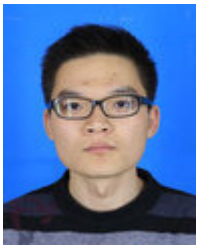
[33] J. An, J. Cheng, X. Gui, W. Zhang, D. Liang, R. Gui, L. Jiang, and D. Liao, "A lightweight blockchain-based model for data quality assessment in crowdsensing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 84–97, 2020.

[34] A. Beniiche, "A study of blockchain oracles," *CoRR*, vol. abs/2004.07140, 2020.

[35] J. Hu, M. J. Reed, N. Thomos, M. F. AI-Naday, and K. Yang, "Securing SDN controlled IoT networks through edge-blockchain," vol. 8, no. 4, pp. 2102–2115, 2021.

[36] D. P. Ballou and H. L. Pazer, "Modeling data and process quality in multi-input, multi-output information systems," *Management science*, vol. 31, no. 2, pp. 150–162, 1985.

[37] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics*, vol. 34, no. 3, pp. 307–318, 1987.



Xi Tao received his B.Eng. degree and M.Eng degree in Electrical Engineering from Xi'an Jiaotong University, Xi'an, China, in 2013 and 2016, respectively. He received his Ph.D. degree in Computer Science from University of New Brunswick, Fredericton, NB, Canada, in 2020. He is a postdoctoral researcher in the Department of Computer Science and Operations Research, University of Montreal, Montreal, QC, Canada. His research interests include mobile crowdsensing, edge computing, and IoT.

Abdelhakim Senhaji Hafid is Full Professor at the University of Montreal. He is the founding director of Network Research Lab and Montreal Blockchain Lab. He is research fellow at CIRRELT, Montreal, Canada. Prior to joining University of Montreal, he spent several years, as senior research scientist, at Bell Communications Research (Bellcore), NJ, US, working in the context of major research projects on the management of next generation networks. Dr. Hafid was also Assistant Professor at Western University (WU), Canada, Research Director of Advance Communication Engineering Center (venture established by WU, Bell Canada and Bay Networks), Canada, researcher at CRIM, Canada, visiting scientist at GMD-Fokus, Germany and visiting professor at University of Evry, France. Dr. Hafid has extensive academic and industrial research experience in the area of the management and design of next generation networks. His current research interests include IoT, Fog/edge computing, Blockchain, and Intelligent Transport Systems.