Fair, Secure and Trustworthy Crowdsensing in Spontaneous Localized Settings

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

Moin Hussain Moti
201501066
moin.moti@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
June 2020

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "**Fair, Secure and Trustworthy Crowd-sensing in Spontaneous Localized Settings**" by **Moin Hussain Moti**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Dr. Sujit Prakash Gujar

To my pollyannaish parents and fabulous friends

Acknowledgments

Just about two years ago, around the same time (May 2018), I was gearing up for my trip to Hong Kong to begin my research internship in Symlab, HKUST. I was a CSE (B.Tech + Honours) student then, and it was my advisor Prof. Sujit Gujar who had suggested I choose the research internship over some corporate one. I thank Prof. Pan Hui (Head of Symlab) for giving me that opportunity. There, I worked under the supervision of Dr. Dimitris Chatzopoulos; to date, those remain the most educational months of my scholarly days. Naturally, I chose to go there next summer as well to continue our work. By the end of my second research internship there, I was convinced to pursue research beyond B.Tech. In the next semester, I converted to a CSD (B.Tech + MSc.) student, and the work we did became the base of this thesis. I am very thankful to my advisor Dr. Sujit Gujar and my mentor Dr. Dimitris Chatzopoulos for guiding me on this scholarly journey over the last few years.

My parents always prioritized my needs above theirs and took care of me in every way possible. There aren't enough words or ways in which I can express my gratitude for everything they have done for me. I hope I can continue making them proud. The last five years were made special because of the company of a few extraordinary people, who I proudly call my friends. Ashwin, Tirth, Megh and Anubhab, you guys have always been there for me, good, bad, and ugly times alike. Food tasted better, and assignments felt easier in your company, what more can a college student ask for? College life would have been super sad if not for you fellas. I can't thank you all enough. Sankarshan and Manisha, my partners in crime, writing papers, and attending conferences, can never be funnier without you guys. I couldn't have asked for better friends cum colleagues. Thank you for bearing with me, mates. Susobhan, you have rescued me countless of times from tricky situations, thank you for saving me from all those troubles and making life easier. Karthik and Sneha, even dull days gets interesting with you guys, thank you for brightening up my bad days.

Lastly, I thank all the staff, administration, and faculties for helping me throughout the college in every and all possible ways.

Abstract

Crowdsensing with mobile agents is a popular way to collect data, especially in the context of smart cities where the deployment of dedicated data collectors is expensive and ineffective. Existing crowdsensing mechanisms are good for general purpose queries but lack any additional benefits for specialized settings. This mechanism focuses on spontaneous location specific queries, we call it *Spontaneous Localized Settings*. We define the required characteristics for this specific setting and evaluate existing mechanisms for the same. Agent participation in good numbers is quintessential for any crowdsensing mechanism to work. Since our setting is location specific and bounded by short time, it allows only agents nearby the query location to participate. Therefore, we emphasize on a *fair* rewarding mechanism and a *secure* and *trustworthy* framework to maximize agent participation. Consequently, fairness in reward dsitribution and trustworthy crowdsensing systems are the two central themes of this thesis.

Peer prediction based mechanisms are widely used in crowdsensing systems to elicit truthful information from agents, however, they often fail to reward the participating agents equitably. Honest agents can be wrongly penalized if randomly paired with dishonest ones. In this work, we introduce *selective* and *cumulative* fairness. We characterize a mechanism as fair if it satisfies both notions and present *FaRM*, a representative mechanism we designed. FaRM is a Nash incentive mechanism that focuses on information elicitation for spontaneous local activities which are accessible to a limited number of agents without assuming any prior knowledge of the event. All the agents in the vicinity observe the same information. FaRM uses *(i)* a *report strength score* to remove the risk of random pairing with dishonest reporters, *(ii)* a *consistency score* to measure an agent's history of accurate reports and distinguish valuable reports, *(iii)* a *reliability score* to estimate the probability of an agent to collude with nearby agents and prevents agents from getting swayed, and *(iv)* a *location robustness score* to filter agents who try to participate without being present in the considered setting. Together, report strength, consistency, and reliability represent a fair reward given to agents based on their reports.

Furthemore, it requires trustworthy framework and to guarantee that the collected data are accurately acquired and reported. Trustworthy and transparent frameworks can be implemented via smart contracts on blockchain to enable robust crowdsensing mechanisms. To achieve this goal, we develop *Orthos*, a blockchain-based trustworthy framework for spontaneous location-based crowdsensing without assuming any prior knowledge about them. We employ game-theoretic mechanisms to incentivize agents to report truthfully and ensure that the information is collected at the desired location while ensuring the privacy of the agents. We evaluate existing mechanisms based on all desirable characteristics for the settings and identify RPTSC as the most suitable option. Orthos implements the RPTSC mechanism using *smart contracts*. Additionally, as location information is exogenous to RPTSC, we design the *Proof-of-Location* protocol to ensure that agents gather the data at the desired locations. We built the decentralized applicaiton based on Orthos protocol and examine its performance on Rinkeby Ethereum testnet and conduct experiments with live audience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Crowdsensing systems have been employed to gather data for a wide variety of applications. A few examples are pollution measurements in cities, product reviews on e-commerce websites, feedback on mobile applications on application markets, working hours and other details of a business, etc. Depending on the type of the information of interest and the available time for reporting, the devices that can collect the data and share them with the crowdsensing systems vary. The focus of this work is on location specific queries that need to be answered within a short time, we call it *spontaneous localized settings*. Such settings are challenging to handle since information regarding the query subjects can only be collected by mobile agents that are located in the area and, possibly, by pre-deployed static sensors. For example, consider a food deli serving various snacks to be the subject of interest and let the query be the availability of a certain snack. Since the food deli is only relevant to the neighborhood and the queries can be specific, formerly Internet services, like Google Neighbourly[1], need to employ mobile agents.

The first main challenge in acquiring information using crowdsensing mechanisms is to ensure that the agents are truthful. An agent, for example, may have some grudge against an agency and therefore reports their service as a lousy experience irrespective of the services of the agency. Agents can be categorized as *honest*, *free-riders* or *misbehaving*. Honest agents are expected to put effort into assessing the query subject and reporting truthful information. Free-riders do not put effort into the assessment and try to earn free credits by guessing the information requested for the subject. Misbehaving agents try to trick the mechanism by colluding among themselves and reporting false information. Crowdsensing mechanisms must be robust against free-riders and misbehaving agents. For that, they incorporate incentive mechanisms to engage agents to submit accurate reports. Rational agents target on maximizing their utility (payments, reputation, etc.) while not sacrificing a substantial amount of resources (battery lifetime, dataplan usage, etc). Thus, we need to offer proper Incentives to report observed signals truthfully. This is called *incentive compatibility* (IC).

For traditional settings, there exist many popular peer prediction based crowdsensing mechanisms inspired by the work of Prelec [29], who proposed the *Bayesian Truth Serum* and Miller *et al.* [25], who

[1]https://edition.cnn.com/2018/11/21/tech/google-india-neighbourly-app/index.html

Where's the best serradura pudding in MGM Macao
Where can I find a salon near Torre de Macao
HMM...

Figure 1.1: Example Queries in Spontaneous Localized Settings

developed *Peer-to-Peer Prediction*. These mechanisms rate an agent's report against the report of another randomly selected agent and if they match, both agents are rewarded based on the prior probability of their answer. However, when considering spontaneous localized settings, it is probable to not have readily available prior knowledge about the query subject. Also, there is a chance of matching an honest agent with a misbehaving one, which results in an unfair evaluation. Although these mechanisms are Bayesian Incentive Compatible, they do not ensure *fairness*. Goel and Faltings [17], recently proposed a fair mechanism for crowdsensing. However, they consider queries where the answers are known and hence, their mechanism is not applicable to spontaneous localized settings. These crowdsensing mechanisms also take for granted that the agents are present at the requested area or assume that there exist parties (e.g., cellular network providers) that can assure the crowdsensing mechanism about the existence of an agent in the required location [9]. In the case where such location verification mechanisms are not assisting crowdsensing mechanisms, agents can abuse the system by faking their location.

Generally, spatio-temporal data for modern applications and services is acquired either by centralized entities (e.g., online reviews about a restaurant) or mobile agents (e.g., current queue length in a coffee shop). In the first case, the entire trust is placed on one single entity which exposes the system to a numerous risks. In the second case, the trust is distributed among several agents, as well as the risk. The need for accurate location-based reports from mobile agents is, among others, highly motivated by advances in smart cities, and more generally, smart infrastructure. Representative examples can be found on health monitoring systems (e.g., pollution levels in specific areas), smart farming, and others. For example, every year crop insurance firms receive numerous claims that need to be verified. The current solution is to send dedicated agents for on-field inspection. Crowdsensing mechanisms can help to incentivize agents to report honestly but the platform for exchanging the information remains a bottleneck in terms of efficacy and reliance. Trustworthy crowdsensing frameworks can reduce the inspection cost by employing mobile agents in the vicinity of the crop plot to verify the claims.

Queries for spatio-temporal data in *spontaneous localized settings* are valid only for a short duration and give limited time to mobile agents to respond. Due to such spontaneous nature of the queries, it is probable for agents to not have readily available prior knowledge. Also, depending on their location, plenty of agents may not be found in the vicinity. The potential unavailability of agents in locations of interest and the lack of prior knowledge motivates the need for secure and trustworthy frameworks that can ensure the quality of the crowdsensed information. Mobile agents are expected to utilize their devices with multiple sensors to support services to *(i)* deploy resources, *(ii)* produce unbiased measurements, *(iii)* augment sparse data collected via static sensors, and *(iv)* supplement missing data caused by malfunctioning static sensors. There are three main challenges to secure and trustworthy acquisition of information in spontaneous localized settings via mobile agents: *(i)* to ensure that they are truthful, *(ii)* to validate their presence in the examined settings where they have limited time to submit their reports, and *(iii)* to preserve their privacy while maintaining the transparency of the process.

1.1 Contributions

1. Firstly, we define the *spontaneous localized settings* and a set of six necessary characteristics required by any mechanism that acquires data via crowdsensing to apply to these settings. We investigate 17 existing crowdsensing mechanisms based on the proposed necessary characteristics and evaluate their applicability to spontaneous localized settings. We identify fairness in reward distribution and the absence of a secure and trustworthy platform for information exchange as two primary challenges to mobile crowdsensing in our settings. We tackle and propose two separate novel solutions to these problems.

2. To ensure fair rewards in our settings, we first formally define two notions of fairness, namely, *selective fairness* and *cumulative fairness*. Selective fairness ensures that any two agents with the same reports are scored similarly for that query. Cumulative fairness ensures that an agent's consistency of truthful reporting is accounted for in the final reward as her reports are more trustworthy, and therefore, valuable to the system.

3. We then propose *FaRM* (Fair Reward Mechanism), a Nash incentive compatible crowdsensing mechanism that incentivizes agents to report truthfully and satisfies selective and cumulative fairness. FaRM is also resistive towards colluding groups of misbehaving agents and therefore makes a perfect case for our settings.

4. We also design the *Proof-of-Location* (PoL) protocol, that does not require any fixed infrastructure to function, to force every mobile agent that wants to submit a report to provide a proof that their location is within a threshold.

5. To provide a secure and trustworthy platform for information exchange in mobile crowdsensing, we create a blockchain based framework *Orthos*.

1.2 Overview

The thesis is organized as follows:

- **Chapter 2** provides background on *mobile crowdsensing*, *Blockchains* and *Smart Contracts*.
- **Chapter 3** introduces *Spontaneous Localized Settings* and discusses various characteristics required by a crowdsensing mechanism to be applicable in it. Additionally, it discusses the shortcomings of various existing crowdsensing mechanisms in the settings.
- **Chapter 4** defines two notions of fairness in the spontaneous localized settings and proposes FaRM, a fair reward mechanism for crowdsensing in the settings. We prove FaRM satisfies the two notions of fairness, is Nash incentive compatible and resists collusion among agents.
- **Chapter 5** proposes *Orthos*, a blockchain based framework for secure and trustworthy crowdsensing in spontaneuos localized settings. We provide a detailed *proof of location* (PoL) protocol for location verification in mobile crowdsensing and discuss the challenges in implementing such a framework over a transparent public-distributed blockchain ledger.
- **Chapter 6** summarizes the contribution of our work and presents ideas of future work.

Chapter 2

Background

In this chapter, we briefly explain the basics of Nash equilibrium, mobile crowdsensing, blockchains and smart contract.

2.1 Mobile Crowdsensing.

Mobile crowdsensing is a paradigm that utilizes the ubiquitousness of mobile users who are carrying smartphones, and can collect and process data. In a typical game setting $\Gamma = \langle \mathcal{A}, (S_i), (u_i) \rangle$, $\mathcal{A}$ represents the total number of agents participating in the game, S_i denotes the signal space of an arbitrary agent i, and u_i denotes agent i's utility function in that game with respect to her reported signals. Each agent observes a signal s_i and reports a signal s_i'. If $s_i = s_i'$, then she is reporting truthfully. The goal of mobile crowdsensing is to elicit truthful information from agents corresponding through mobile devices.

Ra *et al.* [30] developed *Medusa*, a framework to develop crowdsensing applications. However, they employ cloud resources instead of a blockchain and do not guarantee agents' privacy. Han *et al.* [18], motivated by the fact that if the available mobile agents are fewer than the required ones, incentive mechanisms will lose efficacy, proposed *HySense*. It combines mobile devices with static sensor nodes. Furthermore, Wang *et al.* [42] propose *effSense*, an energy-efficient and cost-effective framework to reduce the participation cost of mobile agents.

2.2 Game Theory Essentials

In a game setting motivated by mobile crowdsensing, initially, all agents only have information about the game query and their signal space S_i. Then, an arbitrary agent i observes signal $s_i \in S_i$. Agent i knows this is the true signal and that other agents observed the same. Agent i is free to report any signal (not necessarily same as the observed signal) to the system. We can assume any rational agent to report the signal that maximizes her utility. Under such conditions, it is possible to attain a Nash Equilibrium in the game such that all agents' utility is maximized.

Definition 2.2.1 (Pure Strategy Nash Equilibrium)**.** *Given a strategic form game* $\Gamma = \langle \mathcal{A}, (S_i), (u_i) \rangle$*, the strategy profile* $s^* = (s_1^*, s_2^*, \ldots, s_n^*)$ *is called a Pure Strategy Nash Equilibrium of* Γ *if*

$$s_i^* = \underset{s_i \in S_i}{\operatorname{argmax}}\, u_i(s_i, s_{-i}^*) \ \forall i \in \mathcal{A}$$

That is, each player's Nash equilibrium strategy is a best response to the Nash equilibrium strategies of the other players [28].

Definition 2.2.2 (Nash Incentive Compatibility)**.** *A social choice function* $u : S_1 \times \ldots \times S_n$ *is said to be Nash Incentive Compatible (or truthfully implementable in Nash equilibrium) if the direct revelation mechanism* $\mathcal{D} = ((S_i)_{i \in \mathcal{A}}, u(.))$ *has a pure strategy Nash equilibrium* $\pi^*(.) = (\pi_1^*(.), \ldots, \pi_n^*(.))$ *in which* $\pi_i^*(s_i) = s_i \forall s_i \in S_i, \forall i \in \mathcal{A}$*.*
That is, directly asking the agents to report their types and using this information in $u(.)$ *to get the social outcome will solve both the problems, i.e., preference elicitation and preference aggregation [28].*

2.3 Blockchains

Blockchains is a distributed mechanism for storing data in the form of transactions. Bitcoin[1], Ethereum[2] and Ripple[3] are few notable public-distributed ledgers based on the blockchain architecture. These ledgers are maintained by their global peer-to-peer network of nodes. All transactions are stacked in a block and then the block is appended to the public-ledger. Each block contains a cryptographic hash of the previous block, a timestamp and transaction data. The data is hashed and encoded into a *Merkel Tree*. The cryptographic hash that forms the link to the previous block iteratively goes all the way back to the genesis block, this ensures the integrity of the whole blockchain. The data once recorded on a blockchain ledger is effectively immutable as any moderation would require alteration of all subsequent blocks which requires consensus of majority of the network nodes. Because of the decentralized nature of the blockchain, data is replicated across all nodes of the network. This protects the network from any threats to a particular node.

However, publishing a block is a challenging process and requires a lot of resources, its termed as *mining* in blockchain nomenclature. A miner must validate all the transactions stacked in the block and solve a crytographic puzzle through bruteforce computations in order to mine a block, the solution obtained on solving the puzzle is termed as *proof-of-work*. The time taken to mine a block is variable and depends mainly on the difficulty level of the puzzle. The *block time* is the average time it takes for the network to generate one block in the blockchain. The block time for bitcoin is around 10 mins while the block time on Ethereum is around 15 seconds.

[1]https://bitcoin.org/
[2]https://ethereum.org/
[3]https://ripple.com

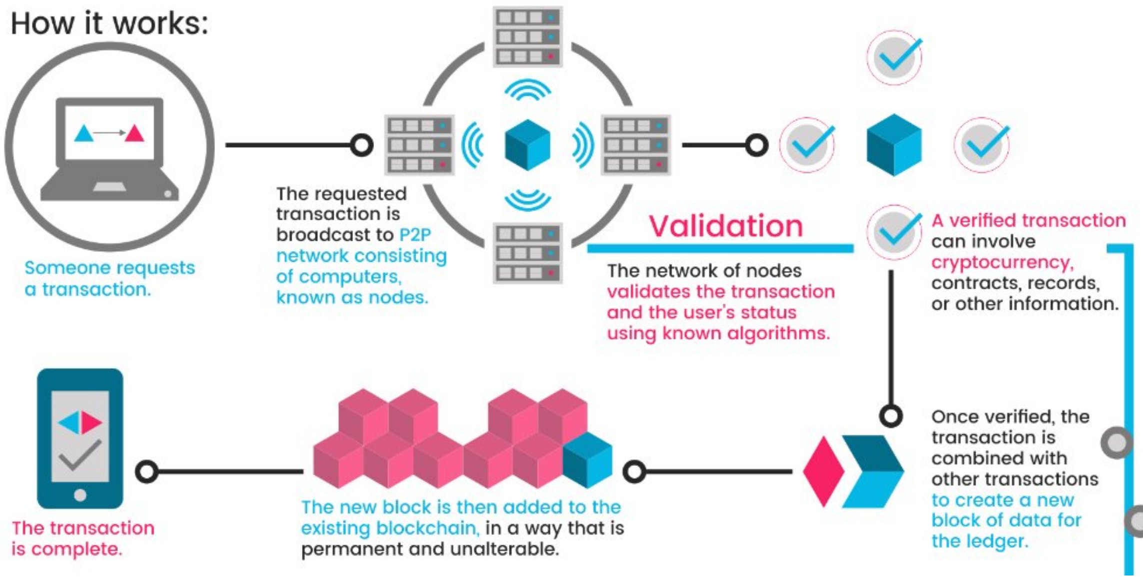

How it works:
Someone requests a transaction.
The requested transaction is broadcast to P2P network consisting of computers, known as nodes.
Validation
The network of nodes validates the transaction and the user's status using known algorithms.
A verified transaction can involve cryptocurrency, contracts, records, or other information.
Once verified, the transaction is combined with other transactions to create a new block of data for the ledger.
The new block is then added to the existing blockchain, in a way that is permanent and unalterable.
The transaction is complete.

Figure 2.1: Blockchain

2.3.1 Smart Contracts.

Nick Szabo [37] first coined the term and proposed the idea of a smart contract, ”a set of promises, specified in digital form, including protocols within which the parties perform on the other promises”. The idea was later adopted by blockchains to offer additional functionalities on the stored data. Each smart contract takes information as an input and processes that information using the set of rules defined in the contract. It can also trigger other smart contracts and access information stored on remote servers. Every smart contract is executed in a virtualized environment maintained by every peer in the blockchain. Whenever a smart contract is called, via a transaction, it is executed when the nodes that maintain the blockchain process the corresponding transaction. Every node has to execute the code of the contract and depending on its complexity and the capabilities of the peers, it may take a lot of time and resources. This contract execution paradigm motivates proposals for off-chain code execution.

Blockchain-based mechanisms can execute parts of their modules on remote servers, also known as *oracles*, to improve their performance and increase the privacy of the agents [13]. Given that everything stored in the blockchain, including the code of smart contracts and the data stored on them, is visible to everyone, private information should be stored on oracles to motivate agents’ participation. By building on top of a blockchain, smart contracts provide a trusted framework for many potential applications. For example, Bogner *et. al.* [3] present a decentralized application for sharing resources like Uber and Airbnb without the involvement of any trusted third party. Internet of Things (IoT) devices form a crucial part of any smart city project, however, privacy and security remain an issue. The authors of [47] and [49] propose smart contract based solutions for safe and secure access control of IoT devices.

Unlike other software applications deployed online, the code of a smart contract cannot be altered once deployed on the network. The authors of [1] compared five different tools for detecting vulnerabilities in the smart contract, namely *Oyente* [24], *Securify* [39], *Remix* [14], *Smartcheck* [38] and *Mythril* [10], tested them against 15 different vulnerabilities and reported Smartcheck to detect the highest number of vulnerabilities. Later, the authors of [16] proposed *Slither*, an open source static analysis framework for smart contracts. Slither reportedly outperformed other tools in terms of performance, accuracy, and robustness.

2.3.1.1 Ethereum Smart Contracts

Ethereum is one of the most popular smart contracts platform. The most recommended language to code smart contracts on Ethereum is Solidity[4]. A smart contract written in high-level-language code is then compiled to bytecodes. This bytecode is published to the Etheruem blockchain where it is executed on Ethereum Virtual Machine (EVM). The EVM is a turing complete machine that consumes resources in the form of *gas* units to execute commands in the smart contract.

[4]https://solidity.readthedocs.io

Chapter 3

Related Work Analysis in Spontaneous Localized Settings

Existing mechanisms for information aggregation have been designed for the two following settings: *(i)* online reviews about products and services, and *(ii)* community sensing regarding prevalent topics like pollution level reporting over a wide region for a considerable duration of time. Additionally, to the best of our knowledge, existing literature on crowdsensing has not discussed information aggregation of location-specific spontaneous tasks on local subjects. Most papers study an *independent value* model where the agents can independently observe varying signals. In this model, the truthful report of agents does not need to be the same. However, spontaneous localized settings follow a *common value* model where all agents observe the same signal.

The spontaneity of the requests and zero prior knowledge about the query subject adds to sophistication of the settings. It, therefore, requires very specific mechanisms that can be used in such scenarios. Below, we list seven essential characteristics a mechanism needs and discuss the applicability of seventeen mechanisms with respect to these characteristics.

3.1 Essential Characteristics

C1) Nash Incentive Compatibility: As ground truth is not readily available in many scenarios, the verification of an agent's report depends on the reports of other agents. Therefore, the mechanism must induce *Nash Equilibrium* where truthful reporting is the best response given other agents are also reporting truthfully.

C2) No common knowledge: The local setting is specifically designed for entities, the information to which is otherwise very difficult to access on internet forums. As a result, common knowledge parameters like *prior belief models* and *posterior expectations* used by most mechanisms are rendered futile for local setting.

C3) Minimalistic Mechanism: We say a mechanism is *minimalistic* if the agents only have to submit the *information report* i.e. observed private signal for the query subject. Many mechanisms require an agent to submit two reports. In addition to the *information report*, these mechanisms require *prediction report*, that reflects the agents' belief about the distribution of information reports in the population. In the local setting, agents have limited time to respond to the request, therefore, we require a *minimalistic* mechanism where agents only have to submit the information report.

C4) Interim Individual Rationality: Aggregated information from a few agents is less reliable and more prone to human error. Hence, increased participation contributes to a more robust information. To ensure that, the mechanism must guarantee non-negative rewards to participating agents. If a mechanism ensures positive expected utility to the agents, it is said to satisfy Interim Individual Rationality (IIR).

C5) Prevent Free-riders: Free-riders can benefit from an IIR mechanism by submitting random responses and hence, the mechanism should not admit *uninformed equilibria* where free-riders benefit by abusing the mechanism.

C6) Collusion Resistant: Agents must be located nearby the query subject in local setting. Agents operating in close proximity exposes the system to collusion. Therefore, the mechanism should be collusion resistant to prevent abuse of the system.

C7) Non-Binary signal space: Binary signal space can compromise the accuracy of information and therefore the employed mechanism should support non-binary signals.

Any mechanism that has above set of attributes can be used in LocalBrain. We now investigate existing mechanisms in the literature to examine their applicability in local settings.

3.2 Existing Mechanisms

3.2.1 M1, by Miller *et al.* [25]

M1 uses an information market setting to elicit information from the agents. This mechanism was mainly intended to be used with product reviews. It heavily relies on the *common knowledge assumption*. For each query there exists a prior distribution of probabilities assigned to all possible responses. Every agent's report is used to update the probability distribution of the report of the future agent. The final score is based on the accuracy of the prediction of an agent's report by the system i.e. comparison between the likelihood assigned to the agent's possible report and actual report. Spontaenous localized settings involve queries which may not have any prior history and the nearby agents can have varying private beliefs. M1 assumes that every agent shares the same prior belief about the event and that the mechanism knows this prior, and hence, cannot be considered for this settings.

3.2.2 M2, by Prelec [29]

M2 proposes Bayesian Truth Serum (BTS), which does not require knowledge of any common prior information. Each agent provides two reports, namely, information report and prediction report. Information report consists of the signal which represents the agent's belief. And prediction report here consists of the prediction of the proportion of agents that will pick the rest of signal choices. Agents recieve information score and prediction score respectively for their report and their final reward is the sum of all the scores.

However, the mechanism is applicable only to large number of agents. Spontaneous localized settings allows only agents that are present in the vicinity of entity in question which may not be large. Moreover, the mechanism is not minimal, does not prevent free-riding or collusion among misbehaving agents. Nevertheless, this seminal work is one of the first to enable information elicitation without common knowledge assumption and inspired several new mechanisms.

3.2.3 M3, by Lambert and Shoham [23]

This mechanism was mainly designed to conduct truthful surveys through crowdsensing. It does not assume any prior common knowledge or even any information structure for the agents. However, in equilibrium, it is indifferent towards misreporting and honest agents, thus allowing agents to free ride. Also, the mechanism does not address collusion among misbehaving agents.

3.2.4 M4 & M5 by Witkowski and Parks [43]

M4 and M5 were originally designed for product reviews using crowdsensing. They do not make any common knowledge assumption and agents can have private beliefs of the product. However, both the mechanisms rely on *temporal separation*. It means, the mechanisms require one report before task completion from each agent (which establishes the prior belief of the agent) and one report after the task completion (establishing posterior belief). In M4, for an agent i, the system rates its report against the belief report it generated for a random agent j. In M5, instead of a belief report generated by thee system, agent i's report is rated against the actual report of agent j.

Firstly, their dependence on the temporal separation makes these mechanisms very slow and non-minimal. Secondly, these crowdsensing mechanisms makes no attempt to prevent free-riding or collusion from misbehaving agents. Because of these reasons, both M4 and M5 are not suited to our settings.

3.2.5 M6 by Witkowski and Parkes [45]

M6 proposes Robust Bayesian Truth Serum (RBTS), which simplifies BTS and does not necessarily require large number of agents. It requires two reports (information and prediction report) but with no reliance on temporal separation. However, the mechanism is only applicable to queries with binary signal space. Having only two options to choose from is sub-optimal and compromises with the quality of aggregated information. Furthermore, the mechanism relies on common prior knowledge which is not available in our settings.

3.2.6 M7 & M8 by Radanovic and Faltings [31], [32]

These two mechanisms are derivatives of RBTS mechanisms that modify the payment scheme to apply to non-binary and continuous signal spaces. Apart from that, they share the advantages and disadvantages of their parent mechanism.

3.2.7 M9 by Zhang and Chen [48]

The mechanisms does not require any prior common knowledge, however, it assumes the existence of a common prior and that it is known to all agents, which may not be possible in our settings. Similar to many mechanisms above, this mechanism also requires both information and prediction report. Notably, one special attribute of this mechanisms is that it does not assume any information structure for agents (i.e. agents can have varying signal spaces). However, this is not generally useful to our settings, so its redundant advantage for this mechanism in our settings. The mechanism also does not prevent free-riding or collusion of misbehaving agents. Overall, it does not meet the requirements of our settings.

3.2.8 M10 by Riley [35]

M10 is the first minimalistic mechanism in our list, it only requires the information report and the prediction report is optional. However, it assumes that all the agents with the same report have the same posterior expectations. Our settings requires a mechanism which does not depend on such assumptions as the requests can arrive almost spontaneously and the mechanism must be robust to incorporate the report of as many nearby agents willing to participate as possible. Moreover, the mechanism does not address free-riding and collusion resistivity against misbehaving agents.

3.2.9 M11 by Jurca and Faltings [21]

M11 was designed for the specific case of generating ratings of various service providers and their clients through crowdsensing. Both service providers and clients are treated equally as agents. The mechanisms deploys an interactive setting where two agents (client agent and service provider agent) rate each other. The payoffs for the agents are derived from *prisoner's dilemma* game matrix. Our setting is clearly very differnet as it involves aggregating responses from many agents and hence this mechanisms is not applicable here.

3.2.10 M12 by Jurca and Faltings [22]

M12 is an improved version of M11 as it allows multiple agents to participate in the review process. It formulates the problem of rewarding agents into a linear and conic optimization problem, where the solution results in a optimal scoring rule that minimizes the usage of budget. However, the mechanisms suffers from the assumption of common prior knowledge and no provisions for preventing free-riding and collusion among misbehaving agents.

3.2.11 M13 by Dasgupta and Ghosh [11]

This is a robust incentivizing mechanism that rewards agents based on the amount of effort they put into answering the query. Therefore, unlike the above mechanisms, it discourages free-riders from participating in the process. However, it requires few trusted reports in order to function properly which cannot be obtained in our settings. Furthermore, the mechanism only supports binary signal space, which limits the mechanism‘s usability in our settings.

3.2.12 M14 by Faltings *et al.* [15]

It is a minimalistic mechanism which requires only information report. However, the mechanism relies on prior common knowledge and that all agents are aware of it. Furthermore, the mechanism also admits uninformed equilibria, where agents do not perform measurements. Such equilibria can result in agents free-riding and lowering the quality of the aggregated information.

3.2.13 M15 by Radanovic and Faltings [33]

M15 improves M14 by introducing logarithmic PTS and eliminating the dependency on prior belief model. The mechanism produces worse payoff than truthful reporting for uninformed equilibria and against misbehaving agents acting on collusion strategies. However, LPTS satisfies all characteristics, however, it is more suitable for high crowd settings. Nevertheless, we describe LPTS below,

LPTS scores a report based on its statistical significance. Consider an agent i submitting report $r_i \in S$ to the system. Let P denote the set of all her *peers* (agents that are in vicinity of agent i) and σ denote the set of all her *reference peers* (agents that are not each other's peers nor peers of agent i). Then, the statistical significance of a report is defined as

$$\frac{X_{local}(r_i)}{X_{global(r_i)}} \tag{3.1}$$

where,

$$X_{local(r_i)} = \frac{1}{|P|} \sum_{p \in P} 1_{r_p} = r_i \tag{3.2}$$

$$X_{global(r_i)} = \frac{1}{|\sigma|} \sum_{q \in \sigma} 1_{r_q} = r_i \tag{3.3}$$

The final reward for agent i is calculated as,

$$score = \log \frac{X_{local}(r_i)}{X_{global(r_i)}} \tag{3.4}$$

3.2.14 M16 & M17 by Radanovic and Faltings [34]

are optimized versions of M15. M16, *Peer Truth Serum for Crowdsourcing* (PTSC) is more robust than M15 in cases where the number of participating agents is small. M17, *Robust Peer Truth Serum for Crowdsourcing* (RPTSC) is very similar to PTSC but furthermore robust as it excludes the possibilities of ill-defined results from PTSC. Both are minimalistic payment mechanisms that incentivize the honest behavior of agents. They are independent of agents' private prior beliefs and satisfy BIC. Agents only announce their observation in the information reports to participate in the process. For every report, they generates a non-negative score for each agent. Any uninformed equilibrium (equilibria where agents do not perform measurements), including random reporting or collusion on one value and collusion strategies that are based on agents' measurements, result in worse payoff than truthful reporting. Thus, agents are incentivized to submit honest reports. PTSC and RPTSC enable the agents to participate in multiple tasks, however, for our purpose we will restrict to single tasks scenarios. We briefly describe PTSC and RPTSC mechanism below. Please note that PTSC and RPTSC only differ in the second stage of reward computation.

	C1	C2	C3	C4	C5	C6	C7
M1 [25]	✓	-	-	✓	-	-	✓
M2 [29]	✓	✓	-	✓	-	-	-
M3 [23]	✓	✓	-	✓	-	-	✓
M4 [43]	✓	✓	-	✓	-	-	-
M5 [44]	✓	✓	-	✓	-	-	-
M6 [45]	✓	✓	-	✓	-	-	-
M7 [31]	✓	✓	-	✓	-	-	✓
M8 [32]	✓	✓	-	✓	-	-	✓
M9 [48]	✓	✓	-	✓	-	-	✓
M10 [35]	✓	-	✓	✓	-	-	✓
M11 [21]	✓	-	✓	✓	-	-	✓
M12 [22]	✓	-	✓	✓	-	-	-
M13 [11]	✓	✓	-	✓	✓	-	-
M14 [15]	✓	-	✓	✓	✓	✓	✓
M15 [33]	✓	✓	✓	✓	✓	✓	✓
M16 [34]	✓	✓	✓	✓	✓	✓	✓
M17 [34]	✓	✓	✓	✓	✓	✓	✓

Table 3.1: Comparison matrix of the 17 examined mechanisms.

Consider an agent i submits a report $r_i \in S$ to the system. Randomly select a peer agent $p \in P$ and let her report be r_p. Calculate the fractional frequency of agent i's report, denoted by R_i as,

$$R_i(p) = \frac{num_{-i}(r_i)}{\sum_{s \in S} num_{-\{i,p\}}(s)} \tag{3.5}$$

where num is the function that counts occurrences of reported values among all the reports.

For PTSC, calculate the reward of agent i as

$$\tau(i,p) = \alpha \cdot (\tau_0(i,p) - 1) \tag{3.6}$$

where,

$$\tau_0(i,p) = \begin{cases} \frac{1}{R_i(p)} & if \quad r_i \neq r_p \\ 0 & if \quad r_i = r_p \end{cases} \tag{3.7}$$

where α is a constant strictly greater than 0.

In RPTSC, the final reward is calculated as,

$$\tau(i,p) = \begin{cases} \alpha \cdot \left(\frac{1_{r_i = r_p}}{R_i(p)} \right) & if \quad R_i(p) \neq 0 \\ 0 & if \quad R_i(p) = 0 \end{cases} \tag{3.8}$$

where $1_{r_i = r_p}$ is indicator variable (equals to 1 if $r_i = r_p$ and 0 otherwise) and α is a constant strictly greater than 0. $R_i()$ is the same as that in PTSC.

Chapter 4

FaRM: A Fair Reward Mechanism for Crowdsensing in Spontaneous Localized Settings

In this chapter, we focus on crowdsensing in spontaneous localized settings and propose a suitable mechanism. Spontaneous localized settings describe crowdsensing requests in local environments about subjects which are accessible by only a small set of people. For example, the food deli scenario mentioned above, the queue length in a local coffee shop, and others. Any query in these settings refers to a query subject (QS) and is valid only for a bounded time. Also, there does not exist a prior distribution for the possible answers about QS. However, in these settings, the observation environment is assumed to be the same for all the agents in the vicinity of QS. This is similar to the output agreement (OA) setting where two agents observe the same signal and are scored based on the degree of similarity in their reports. OA setting was first introduced by von Ahn and Dabbish [40] and later studied experimentally by Robertson *et al.* [36], and Huang and Fu [19]. Waggoner and Chen [41] were the first to provide a theoretical analysis of the output agreement setting. We introduce spontaneous localized settings in more detail in Chapter 3.

We leverage on the characteristic of a common observation environment to propose the *fair reward mechanism (FaRM)*, a mechanism that incentivizes agents to report honestly, marginalizes free-riders and misbehaving agents and ensures fairness in reward distribution without the need for a known prior distribution. FaRM is a novel Nash incentive compatible mechanism for crowdsensing in spontaneous localized settings. Additionally, we introduce the notions of *selective fairness* and *cumulative fairness*. A mechanism satisfies selective fairness in cases where for two different agents that have submitted the same reports, they are evaluated similarly. Thus, all honest agents receive the same reward if they submitted the same report. We propose *report strength* as one component of the reward scheme integrated in FaRM to ensure selective fairness. Existing peer-prediction mechanisms consider every agent's report equally valuable. FaRM, in contrast to this assumption, considers more valuable the agents with long history of submitting truthful reports compared to agents with lousy history of reporting due to their behavioral tendency to maintain the streak. Cumulative fairness mandates that a fair mechanism should acknowledge an agent's history of submission and consider her consistency as part of final reward. We propose *consistency score* as the second component of FaRM's reward scheme which defines an agent's consistency so far in the system and ensures cumulative fairness. This also helps FaRM minimize the noise caused by false reports by considering the value of a report based on its reporter's consistency. FaRM is designed to ensure selective fairness and cumulative fairness.

Also, since spontaneous localized settings focus crowdsensing related to local query subjects, many participating agents will be located around the vicinity of the subject. Hence, it becomes imperative that the mechanism needs to be collusion resistant. To ensure this, we propose the *reliability score* to estimate agents probability of collusion with nearby agents. This score is designed to prevent the agents from colluding with nearby agents since it directly affects their final reward. In summary, the main highlights of this chapter are[1]:

1. FaRM is Nash incentive compatible mechanism that ensures truthful reporting from agents and guarantees non-negative rewards to all agents (Theorem 4.6.1).

2. FaRM incorporates a payment scheme composed of: *report strength*, *consistency score* and *reliability score*.

3. FaRM uses report strength to ensure selective fairness by distributing the same immediate reward to all the agents with the same report (Section 4.3).

4. FaRM employs consistency score as an estimate of agent's consistency in the system so far. It gets updated after each submission. It ensures cumulative fairness by rewarding the agent for her streak of honest submissions to the system (Section 4.4).

5. FaRM integrates reliability score as an estimate of agent's collusion with nearby agents (Section 4.4.3).

6. FaRM filters reports from agents who are not close to the query subject by incorporating a *location robustness score* (Section 4.5).

4.1 Problem Formulation

Considering a query subject QS, a set of nearby agents $\mathcal{A}$ that choose to participate and asses QS, and a budget $B \in \mathbb{R}$, we want to estimate a function f. For example, QS can be a food deli and f the number of eateries available for purchase. S denotes the signal space for the query and is allowed to be non-binary. Every query in spontaneous localized settings focuses on only one task. The observation area is considered to be in the close vicinity of the QS. Since the query requires to be answered spontaneously, we assume the QS to not change its status for the time period the query remains active. Every agent $i \in \mathcal{A}$ observes the same signal $s \in S$ and reports a signal $r_i \in S$. r_i can be different to s. We define a general utility function for agent i as $\hat{u}_i(r_i, r_{-i})$ where r_i represents agent i's reported signal and r_{-i} $(= (r_1, ..., r_{i-1}, r_{i+1}, ..., r_{|\mathcal{A}|}))$ represents all other agents' reported signal. Note that we assume

[1]The preliminary version of this paper was accepted and presented aet IJCAI'19[27]

Symbol	Description
QS	Query Subject
$\mathcal{A}$	Set of agents participating in the process.
$\mathcal{S}$	Set of all the signals.
s	Observed signal by all agents.
r_i	Reported signal by agent i.
u_i	Reward of agent i.
B	Budget per crowdsensing query.
l_i	Context of a agent.
α_i	Consistency score of agent i.
β_i	Reliability score of agent i.
γ_i	Location robustness score of agent i.
$\mathcal{I}_i$	Nearby agents (internal peers) of agent i.
$\mathcal{E}_i$	External peers of agent i.
$\Phi()$	Strength function for a report type.
φ_1	Strength of strongest report.
φ_2	Strength of second strongest report.

Table 4.1: Notation Table.

$|\mathcal{A}| \geq 3$, i.e. at least 3 agents are participating for a particular query. Also, the sum of total utility (u_i) of all agents should not exceed the budget of the query i.e. $\sum_{u_i} \leq B$.

In the following sections, we discuss the properties of FaRM.

4.2 FaRM: Fair Reward Mechanism

The payment scheme designed for FaRM is the product of three sub-utility functions, namely *report strength*, *consistency score* and *reliability score*. The *report strength* is calculated at the end of every report collection process based on the submitted reports. The *consistency score* is updated for each agent after every participation. For a new agent i, her consistency score, (α_i), is initialized to 0. FaRM also computes a *reliability score* for each agent which is an estimate of an agent's collusion with nearby agents. The consistency score motivates agents to participate and report truthfully while the reliability score prevents agents from colluding with her neighboring agents. We discuss these metrics in Section 4.3, Section 4.4 and Section 4.4.3 respectively where we analyze and prove that truthful reporting forms pure strategy Nash equilibrium (PSNE) for each metric.

4.3 Report Strength

We define the strength of a report as the count of agents who have reported the same signal as agent i. If there is a wide spectrum of possible signals, we discretize the range in buckets and consider two signals as equal, if they belong in the same bucket. Report strength is the first of the three scores we compute for a report. It only depends on the current performance of the agent and hence can be considered as the immediate reward for an agent in a particular query.

4.3.1 Computation of Report Strength

Every $s \in S$ represents a report type and each report type receives certain number of reports. Let Φ be a function that counts the number of reports of the same type. Φ_{r_i} is defined as the strength of agent i's report by measuring how many agents are in agreement with agent i.

$$\Phi(r_i) = \sum_{j \in \mathcal{A}} 1_{r_i = r_j} \tag{4.1}$$

$\Phi(\cdot)$ is a function on a generic report type and is independent of the agent. Thus, for computational efficiency, strength score should be pre-computed for every signal which can be done by maintaining a counter for every signal while iterating over all agent reports. That is for n agents, total computational complexity is $O(n)$. Individual strength score for any agent can then be obtained by just referring to the strength score of the corresponding signal reported.

Observation 4.3.1. *The report strength of the report of an agent is always positive.*

Proof. For any agent i, $\Phi(r_i)$ compares r_i with all the reports including r_i itself, hence ensuring that the report strength is at least 1.

$$i \in \mathcal{A} \Rightarrow \Phi(r_i) \geq 1$$

□

Lemma 4.3.2. *Consider a game induced by u^Φ where $u^\Phi = \Phi$ is the sub-utility function corresponding to the report strength (Φ); and let $\hat{s}$ be the observed signal by all agents and r_i be the report submitted by any agent i. Then under Nash equilibrium,*

$$u_i^\Phi(r_i = \hat{s}, r_{-i} = \hat{s}) > u_i^\Phi(r_i \neq \hat{s}, r_{-i} = \hat{s}) \forall r_i \in S, \forall i \in \mathcal{A}$$

That is, if all other agents were to report truthfully, the best response for agent i in order to maximize her sub-utility u_i^Φ is also to report truthfully.

Proof. Let $\hat{s}$ be the observed signal by all agents and let r_i be the reported signal of agent i. As per Nash equilibrium, assuming every other agent reports honestly i.e. $r_{-i} = \hat{s}$, the strength of agent i's report is given by,

$$\Phi(r_i) = 1 + (|\mathcal{A}| - 1) \times 1_{r_i = s}$$

Consequently, sub-utility function u^Φ can be written as,

$$\begin{aligned} u_i^\Phi(r_i \neq \hat{s}, r_{-i} = \hat{s}) &= 1 \\ u_i^\Phi(r_i = \hat{s}, r_{-i} = \hat{s}) &= |\mathcal{A}| \end{aligned}$$

Hence, truthful reporting i.e. $r_i = \hat{s}$ is the best response for agent to maximize u_i^Φ when all other agents are also reporting truthfully. □

4.3.2 Selective Fairness

Crowdsensing involves evaluating reported signals from agents. It is difficult to evaluate an agent since there is no ground truth available in most scenarios. Existing peer prediction based mechanisms discussed in Section 3.2 rate an agent's report against the report of a random peer agent p_i. This, however, exposes agent i to unfair evaluation because if the peer agent p_i is dishonest or misbehaving and submits false report, it will reflect in worse reward for agent i even if she submits truthful report. [12] state that an algorithm is fair if it generates similar results for agents with similar attributes. This is not possible with peer prediction based mechanisms.

Existing mechanisms in the literature mainly discuss settings which follow an *independent-value model* where agents observe varying signals for the same query. This makes it difficult to rate all agents against equivalent reports. In our case, we follow a *common-value model* where all agents observe the same signal. This makes it easier to rate all agents against the same outcome.

Definition 4.3.3 (Selective Fairness)**.** *Let $\mathcal{A}$ be a set of agents in the vicinity of subject QS and agents $i, j \in \mathcal{A}$ be any two arbitrary agents who submit two identical reports r_i and r_j such that $r_i = r_j$. The utility function of a payment scheme $\tilde{u}$, admits* selective fairness *if,*

$$\tilde{u}_i(r_i, r_{-i}) = \tilde{u}_j(r_j, r_{-j}), \forall i, j \in \mathcal{A}$$

Claim 4.3.4. *Report strength is a selectively fair computation score for agents.*

Proof. Strength $\Phi(.)$ is a function of a generic report type (i.e. signal) and for a generic signal s, it is given as,

$$\Phi(s) = \sum_{j \in \mathcal{A}} 1_{s = r_j} \tag{4.2}$$

Hence, by construction, agents with similar report will share similar strength score and so Φ is selectively fair. $\square$

4.4 Consistency Score (α)

The report of an agent with a history of reporting truthfully is more valuable to FaRM than the report of an agent with a lousy history of reporting. Therefore, instead of distributing rewards based only on the performance of agents in the last query, we keep a metric of every agent's performance so far in the mechanism and take it into consideration while paying the agents. We call this metric *consistency score*.

Consistency score depends on the strength of agent i's report. After every report of agent i, the FaRM updates agent's consistency score. The score increases for accurate reporting and gets penalized for inaccurate reporting. It should be noted, however, that the accuracy is relative since the ground truth is not known and the highest reported signal is used as a proxy for ground truth. Consistent agents receive higher rewards than less consistent ones for the same report submitted to the system because of the higher value of their report. We, therefore, motivate agents to maintain the quality of their reports in order to maintain high rewards. This also encourages agents who have a higher rate of submitting truthful reports to the system to continue participating and at the same time discourages free-riders and dishonest reporters.

4.4.1 Computation of Consistency Score

The consistency score is incremented with respect to the second highest reported signal (denoted by ϕ_2) and is decremented based on the highest reported signal (denoted by ϕ_1).

$$\varphi_1 = \max_{s \in S}(\Phi(s)) \tag{4.3}$$

$$\varphi_2 = \begin{cases} \max 2_{s \in S}(\Phi(s)) & \text{if} \quad \max 2_{s \in S}(\Phi(s)) > 0 \\ \\ \frac{\varphi_1^2 - 1}{\varphi_1} & \text{if} \quad \max 2_{s \in S}(\Phi(s)) = 0 \end{cases} \tag{4.4}$$

where $\max 2$ gives the second highest value and $\varphi_1 \geq \varphi_2 > 0$. φ_1 and φ_2 can be pre-computed before computing individual agent scores. For n agents, the complexity is $O(n)$. Only agents with the strongest reports will have their consistency score incremented. However, it is not enough to submit one of the strongest reports, the report type must hold percentage strictly greater than other report types. Other agents will have their consistency score decremented. It is updated as follows,

$$\alpha_i^t = \begin{cases} \alpha_i^{t-1} - \frac{\alpha_i^{t-1}}{k} \times \frac{(\varphi_1 - \Phi(r_i))}{|\mathcal{A}|} & \text{if} \quad \Phi(r_i) < \varphi_1 \\ \\ \alpha_i^{t-1} + \frac{1 - \alpha_i^{t-1}}{k} \times \frac{(\varphi_1 - \varphi_2)}{|\mathcal{A}|} & \text{if} \quad \Phi(r_i) = \varphi_1, \end{cases} \tag{4.5}$$

where $k \geq 1$. Note that k is just an arbitrary constant which can be used to tweak the reward rate. t represents the sequence of rounds agent i has participated. Once φ_1 and φ_2 are pre-computed, updating α has constant computational complexity per agent (i.e., the overall complexity is $O(n)$).

Claim 4.4.1. *Consistency score is bounded in the range [0,1).*

Proof. Consistency score for any agent is initialized at 0. The score remains 0 until the agent report coincides with the highest reported signal for the first time. Let

$$\alpha_i^{t-1} \in [0,1) \tag{4.6}$$

Case 1: $\Phi(r_i) < \varphi_1$

$$\alpha_i^t = \alpha_i^{t-1}(1 - \frac{\varphi_1 - \Phi(r_i)}{|\mathcal{A}|}) \tag{4.7}$$

Also, we know the following,

$$\Phi(r_i) < \varphi_1 < |\mathcal{A}| \tag{4.8}$$

$$k \geq 1 \tag{4.9}$$

Using Equations (4.6), (4.8) and (4.9) in Equation (4.7), we get,

$$\alpha_i^t \geq 0 \tag{4.10}$$

$$\alpha_i^t < \alpha_i^{t-1} < 1 \tag{4.11}$$

Case 2: $\Phi(r_i) = \varphi_1$

$$\alpha_i^t = \alpha_i^{t-1} + \frac{1 - \alpha_i^{t-1}}{k} \times \frac{(\varphi_1 - \varphi_2)}{|\mathcal{A}|} \tag{4.12}$$

Let $a = \frac{\varphi_1 - \varphi_2}{k|\mathcal{A}|}$ and we already know the following,

$$0 < \varphi_2 \leq \varphi_1 \leq |\mathcal{A}| \tag{4.13}$$

$$k \geq 1 \tag{4.14}$$

From Equations (4.13) and (4.14), we can say

$$a \in [0,1) \tag{4.15}$$

We now rewrite Equation (4.12) as,

$$\alpha_i^t = \alpha_i^{t-1} + (1 - \alpha_i^{t-1}) \cdot a \tag{4.16}$$

Using Equations (4.6) and (4.15) in Equation (4.16), we can say,

$$\alpha_i^t \geq \alpha_i^{t-1} \geq 0 \tag{4.17}$$

$$\alpha_i^t < 1 \tag{4.18}$$

From Equations (4.10), (4.11), (4.17) and (4.18), we prove our claim that $\alpha(\cdot) \in [0, 1)$

□

Notice that the consistency score increases rapidly when it is closer to zero and it decreases fast when it is closer to one. This property motivates new agents to join the system and prevent existing agents from downgrading the quality of their reports.

Lemma 4.4.2. *Consider a game induced by u^α where $u^\alpha = \alpha$ is the sub-utility function corresponding to consistency score (α); and let $\hat{s}$ be the observed signal by all agents and r_i be the report submitted by any agent i. Then considering the property of Nash equilibrium,*

$$u_i^\alpha(r_i = \hat{s}, r_{-i} = \hat{s}) > u_i^\alpha(r_i \neq \hat{s}, r_{-i} = \hat{s}) \forall r_i \in S, \forall i \in \mathcal{A}$$

That is, if all other agents were to report truthfully, the best response for agent i in order to maximize her sub-utility u_i^α is also to report truthfully.

Proof. Let $\hat{s}$ be the observed signal by all agents and let r_i be the reported signal by agent i. We assume every other agent to report truthfully as per Nash equilibrium, i.e. $r_{-i} = \hat{s}$. Then, φ_1 and φ_2 are given as,

$$\begin{aligned} \varphi_1 &= |\mathcal{A}| - 1_{r_i \neq s} \\ \varphi_2 &= \begin{cases} 1 & \text{if} \quad r_i \neq s \\ \frac{|\mathcal{A}|^2 - 1}{|\mathcal{A}|} & \text{if} \quad r_i = s \end{cases} \end{aligned}$$

Using above simplifications, we derive simplified consistency score for agent i as,

$$\alpha_i^t = \begin{cases} \alpha_i^{t-1} - \frac{\alpha_i^{t-1}}{k} \times \frac{(|\mathcal{A}| - 2)}{|\mathcal{A}|} & \text{if} \quad r_i \neq s \\ \\ \alpha_i^{t-1} + \frac{1 - \alpha_i^{t-1}}{k} \times \frac{(1/|\mathcal{A}|)}{|\mathcal{A}|} & \text{if} \quad r_i = s \end{cases}$$

Consequently, the sub-utility function u_i^α can be written as,

$$\begin{aligned} u_i^\alpha(r_i \neq \hat{s}, r_{-i} = \hat{s}) &= \alpha_i^{t-1} - \frac{\alpha_i^{t-1}}{k} \times \frac{(|\mathcal{A}| - 2)}{|\mathcal{A}|} \\ u_i^\alpha(r_i = \hat{s}, r_{-i} = \hat{s}) &= \alpha_i^{t-1} + \frac{1 - \alpha_i^{t-1}}{k} \times \frac{(1/|\mathcal{A}|)}{|\mathcal{A}|} \end{aligned}$$

Given $|\mathcal{A}| \geq 3$ and $\alpha^{t-1} \in [0, 1)$ (from Claim 4.4.1),

$$u_i^\alpha(r_i = \hat{s}, r_{-i} = \hat{s}) > u_i^\alpha(r_i \neq \hat{s}, r_{-i} = \hat{s})$$

Hence, truthful reporting i.e. $r_i = \hat{s}$ is the best response for agent to maximize u_i^α when all other agents are also reporting truthfully. □

4.4.2 Cumulative Fairness

Report from an agent who is consistently reporting truthful information is more valuable than the report of an agent with a lousy history of reporting.

Example: In her summer vacation, Anna joins professional cooking classes out of interest. She likes to frequently practice recipes at home and get reviews from her family and neighbors. Once she makes Turkish delights and has Barbara and Chris independently assess the delicacy. However, they submit contrasting reviews. But Anna recalls Barbara's previous reports and finds her to be misreporting on various occasions in order to make her happy, while Chris has been almost always critical but truthful in his reports. Thus, based on the history of reports of Barbara and Chris, Anna identifies Chris' reports to be more useful than that from Barbara.

It's only fair to take the value of an agent's report into consideration while distributing reward. Motivated by similar analogy, we incorporated consistency score (α) in our final reward structure.

Definition 4.4.3 (Cumulative Fairness). *Let $\mathcal{A}$ be a set of agents in the vicinity of local subject QS and agents $i, j \in \mathcal{A}$ be any two arbitrary agents submitting report r_i and r_j such that $r_i = r_j$. Then the utility function $\hat{u}(.)$ of an arbitrary payment scheme is said to admit cumulative fairness if,*

$$\hat{u}_i^t(r_i = \hat{s}, r_{-i}, \hat{u}_i^{t-1}) > \hat{u}_j^t(r_j = \hat{s}, r_{-j}, \hat{u}_j^{t-1})$$
$$\forall i, j \in \mathcal{A} \ s.t. \ \hat{u}_i^{t-1} > \hat{u}_j^{t-1}$$

Claim 4.4.4. *Consistency score is a cumulatively fair computation score for agents.*

Proof. Let $\mathcal{A}$ be a set of agents in the vicinity of local subject QS and agents $i, j \in \mathcal{A}$ be any two arbitrary agents submitting report r_i and r_j such that $r_i = r_j$ and $\alpha_i^{t-1} > \alpha_j^{t-1}$.

Since there reports are same and strength ($\Phi(.)$) is a selectively fair computation score (from Claim 4.3.4), we can say, $\Phi(r_i) = \Phi(r_j)$.

Case 1: $\Phi(r_i = r_j) < \varphi_1$

$$\alpha_i^t = \alpha_i^{t-1} - \frac{\alpha_i^{t-1}}{k} \times \frac{(\varphi_1 - \Phi(r_i))}{|\mathcal{A}|}$$
$$\alpha_j^t = \alpha_j^{t-1} - \frac{\alpha_j^{t-1}}{k} \times \frac{(\varphi_1 - \Phi(r_j))}{|\mathcal{A}|}$$

Let $a_1 = \frac{(\varphi_1 - \Phi(r_i))}{|\mathcal{A}|} = \frac{(\varphi_1 - \Phi(r_j))}{|\mathcal{A}|}$.

Then, we can rewrite α_i, α_j as,

$$\alpha_i^t = \alpha_i^{t-1}(1 - \frac{a_1}{k}) \tag{4.19}$$
$$\alpha_j^t = \alpha_j^{t-1}(1 - \frac{a_1}{k}) \tag{4.20}$$

Also, we know the following:

$$\Phi(r_i = r_j) < \varphi_1 < |\mathcal{A}| \Rightarrow a_1 < 1 \tag{4.21}$$
$$k \geq 1 \tag{4.22}$$
$$\alpha_i^{t-1} > \alpha_j^{t-1} \tag{4.23}$$

Using Equations (4.21) to (4.23) in Equations (4.19) and (4.20), we get,

$$\alpha_i^t > \alpha_j^t \tag{4.24}$$

Case 2: $\Phi(r_i = r_j) = \varphi_1$

$$\alpha_i^t = \alpha_i^{t-1} + \frac{1 - \alpha_i^{t-1}}{k} \times \frac{(\varphi_1 - \varphi_2)}{|\mathcal{A}|}$$
$$\alpha_j^t = \alpha_j^{t-1} + \frac{1 - \alpha_j^{t-1}}{k} \times \frac{(\varphi_1 - \varphi_2)}{|\mathcal{A}|}$$

Let $a_2 = \frac{(\varphi_1 - \varphi_2)}{|\mathcal{A}|}$.

Then, we can rewrite α_i, α_j as,

$$\alpha_i^t = \alpha_i^{t-1}(1 - \frac{a_2}{k}) + \frac{a_2}{k} \tag{4.25}$$
$$\alpha_j^t = \alpha_j^{t-1}(1 - \frac{a_2}{k}) + \frac{a_2}{k} \tag{4.26}$$

Also, we know the following:

$$0 < \varphi_2 \leq \varphi_1 \leq |\mathcal{A}| \Rightarrow a_2 < 1 \tag{4.27}$$

$$k \geq 1 \tag{4.28}$$

$$\alpha_i^{t-1} > \alpha_j^{t-1} \tag{4.29}$$

Using Equations (4.27) to (4.29) in Equations (4.25) and (4.26), we get,

$$\alpha_i^t > \alpha_j^t \tag{4.30}$$

From Equations (4.24) and (4.30) and Definition 4.9, we say that consistency score admits cumulative fairness. □

4.4.3 Reliability Score (β)

Reliability score provides incentives to agent i to not collude with her nearby agents $\mathcal{I}_i$. It is defined as the ratio of *external agreement* by *internal agreement*. The internal agreement is the percentage of nearby agents (formally identified as *internal peers* $\mathcal{I}_i$) that share the same report as agent i and the external agreement is the percentage of agents other than nearby agents (formally addressed as *external peers* $\mathcal{E}_i$) who share the same report as agent i.

Definition 4.4.5 (Internal and External Peers)**.** *The* internal *peers of agent i (denoted by $\mathcal{I}_i$) are the ones with which i can exchange a message using a device-to-device communication protocol like Wi-Fi direct or Bluetooth. The internal peers combined with the agent's location compose her context l_i. Every agent j that is not directly accessible from i (i.e., $j \notin \mathcal{I}_i$) belongs to her* external *peers denoted by $\mathcal{E}_i$.*

4.4.4 Computation of Reliability Score

For agent i, her reliability score is computed as:

$$\beta_i = \frac{\frac{\left(\sum_{j\in\mathcal{E}_i} 1_{r_i=r_j}\right)}{|\mathcal{E}_i|}}{\frac{\left(\sum_{j\in\mathcal{I}_i} 1_{r_i=r_j}\right)}{|\mathcal{I}_i|} + 1} \tag{4.31}$$

For n agents, the computational complexity for calculating β is $O(n)$ per agent.

Claim 4.4.6. *Reliability score is bounded in the range [0,1].*

Proof. Let

$$b_1 = (\sum_{j \in \mathcal{E}_i} 1_{r_i = r_j}) \;/\; |\mathcal{E}_i| \tag{4.32}$$

$$b_2 = \left[(\sum_{j \in \mathcal{I}_i} 1_{r_i = r_j}) \;/\; |\mathcal{I}_i| \right] + 1 \tag{4.33}$$

By construction, $b_1 \in [0, 1]$ and $b_2 \in [1, 2]$

Hence, $\beta_i = \frac{b_1}{b_2} \in [0, 1]$ □

Lemma 4.4.7. *Consider a game induced by u^β where $u^\beta = \beta$ is the sub-utility function corresponding to reliability score (β); and let $\hat{s}$ be the observed signal by all agents and r_i be the report submitted by any agent i. Then considering the property of Nash equilibrium,*

$$u_i^\beta(r_i = \hat{s}, r_{-i} = \hat{s}) > u_i^\beta(r_i \neq \hat{s}, r_{-i} = \hat{s}) \forall r_i \in S, \forall i \in \mathcal{A}$$

That is, if all other agents were to report truthfully, the best response for agent i, in order to maximize her sub-utility u_i^β, is also to report truthfully.

Proof. Let $\hat{s}$ be the signal observed by all agents and r_i be the reported signal of agent i. We assume every other agent to report truthfully as per Nash equilibrium. Then, the reliability score for agent i's report can be simplified as,

$$\begin{aligned} \beta_i &= \frac{1_{r_i = \hat{s}}}{1_{r_i = \hat{s}} + 1} \\ &= 0.5 \times 1_{r_i = \hat{s}} \end{aligned}$$

Hence, agent i maximizes her reliability score by reporting truthfully, i.e. $r_i = \hat{s}$. □

Claim 4.4.8. *Reliability score prevents agents from colluding with nearby agents.*

Proof Sketch. Reliability score (β) can be seen as,

$$\beta = \frac{\text{external agreement}}{\text{internal agreement} + 1}$$

where external agreement is the percentage of the external peers that agree with the agent and internal agreement is the percentage of the internal peers that agree with the agent. For the sake of convenience we use variables l_i and g_i as,

$$\begin{aligned} l_i &= \text{internal agreement} = \frac{1}{|\mathcal{I}_i|} \sum_{j \in \mathcal{I}_i} 1_{r_i = r_j} \\ g_i &= \text{external agreement} = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} 1_{r_i = r_j} \end{aligned}$$

Let's consider a malicious colluding agent m. We assume agents around QS are reasonably scattered and that the majority of agents are honest. In that case, agent m will have high internal agreement and less external agreement i.e. $l_m > g_m$. Since, $\beta_m \propto g_m$ and $\beta_m \propto 1/(l_m+1)$, the more agent m colludes with internal peers the lower her reliability score drops. In case where nearby agents are also honest, it is still in favor of agent m to submit truthful report since she will want to maintain her reporting agreement with external peers.

On the other hand, if agent m were to be honest while her internal peers collude, the outcome is $l_m < g_m$. This leads to increase in reliability score. Hence, if internal peers of agent m are colluding, it is profitable for agent m to remain honest. Thus, reliability score prevents agents from getting swayed by nearby agents (internal peers). □

4.5 Location Robustness Score (γ)

Apart from the aforementioned scores, we introduce the location robustness score of agent i, γ_i, to detect whether the location of a mobile agent is close to the location of the query subject. Every mobile agent can exchange messages with her internal peers in order to justify that she is within a distance from the query subject [7]. In detail, considering that the context of agent i, l_i is composed of her location l_i^{Loc} and her nearby agents $\mathcal{I}_i$, agent i can send a message to every node $j \in \mathcal{I}_i$ with l_i^{Loc} in it using a communication technology like Bluetooth or WiFi-direct. Every agent $j \in \mathcal{I}_i$ will calculate the distance with agent i, D_{ji} and will respond to the message by including her location l_j^{Loc} and the calculated distance. Agent j can sign her message with a private key in order to make sure that agent i will not alternate her estimation of the difference between the two locations. Based on these messages, agent i will calculate the robustness of her location.

$$\gamma_i = \frac{1}{|\mathcal{I}_i|} \sum_{j \in \mathcal{I}_i} 1_{D_{ji} \leq D^T} \tag{4.34}$$

Which is practically the fraction of their neighbors who are within a distance D^T that depends on the communication technology. For example, two mobile agents can communicate with Bluetooth if $D^T < 20$ meters and with WiFi-direct if $D^T < 50$ meters. Depending on the query subject and the predetermined allowed distance from which a mobile agent can produce a report, D^T may be smaller than the values achieved by the employed communication technology. FaRM uses location robustness to detect fraudulent agents.

Every participating mobile agent, has to submit γ_i together with her report. Given that a fraudulent agent is not able to alternate γ_i, since every message from her internal peers is signed, FaRM filters reports with low γ values.

4.6 Rewards

Once FaRM filters reports using the location robustness score, it updates consistency and reliability scores for each agent, then finally it calculates every agent's reward for that query. The location robustness score (γ) is only used for filtration and not considered in the final reward. The final reward of agent i depends on the product of $\Phi(r_i), \alpha_i$ and β_i. The strength of agent's report factors in the immediate reward for the respective query. The consistency score takes into account agent's tendency to report truthfully and therefore motivate a more consistent agent to participate more. The reliability score keeps the agent from colluding with other agents. An agent who has a history of colluding with nearby agents will have less motivation to participate in the system, hence preventing noise generated by misleading reports. The final utility score (u_i) is computed as:

$$u_i(r_i, r_{-i}) = \frac{u_i^{\Phi}(r_i, r_{-i}) \;.\; u_i^{\alpha}(r_i, r_{-i}) \;.\; u_i^{\beta}(r_i, r_{-i})}{|\mathcal{A}|^2} \times B, \tag{4.35}$$

where B is the reward budget per query. Also, since final utility is the product of all three scores, even if some closely located agents collude in order to increase report strength, they risk decreasing reliability score which can mask the benefits of collusion and result in even worse payoffs.

Theorem 4.6.1. *FaRM is Nash incentive compatible with guaranteed non-negative payoffs and weak budget balanced.*

Proof. Let $\hat{s}$ be the signal observed by all agents and r_i be the reported signal of agent i. We assume all other agents to report truthfully as per Nash equilibrium i.e. $r_{-i} = \hat{s}$.
Let

$$w_i(r_i, r_{-i}) = u_i^{\Phi}(r_i, r_{-i}) \;.\; u_i^{\alpha}(r_i, r_{-i}) \;.\; u_i^{\beta}(r_i, r_{-i})$$

and

$$W_{-i}(r_i, r_{-i}) = \sum_{j \in \mathcal{A} \setminus i} w_j(r_j, r_{-j}),$$

then we can rewrite u_i as:

$$u_i = \frac{w_i}{|\mathcal{A}|^2} \times B$$

For a fixed budget B and $|\mathcal{A}|$ number of agents participating,

$$argmax_{r_i}(u_i) \Rightarrow argmax_{r_i}(w_i)$$

Also, w_i can be written for Nash equilibrium case as,

$$\begin{aligned}
w_i(r_i \neq \hat{s}, r_{-i} = \hat{s}) =& u_i^{\Phi}(r_i \neq \hat{s}, r_{-i} = \hat{s}) \ . \\
& u_i^{\alpha}(r_i \neq \hat{s}, r_{-i} = \hat{s}) \ . \ u_i^{\beta}(r_i \neq \hat{s}, r_{-i} = \hat{s}) \\
w_i(r_i = \hat{s}, r_{-i} = \hat{s}) =& u_i^{\Phi}(r_i = \hat{s}, r_{-i} = \hat{s}) \ . \\
& u_i^{\alpha}(r_i = \hat{s}, r_{-i} = \hat{s}) \ . \ u_i^{\beta}(r_i = \hat{s}, r_{-i} = \hat{s})
\end{aligned}$$

Using Lemma 4.3.2, Lemma 4.4.2 and Lemma 4.4.7, we can say $w_i(r_i = \hat{s}, r_{-i} = \hat{s}) > w_i(r_i \neq \hat{s}, r_{-i} = \hat{s})$

Hence, agent i gains maximum utility when she reports truthfully i.e. $r_i = s$.

Also since, $\Phi(r_i) > 0$ (from Observation 4.3.1), $\alpha_i \in [0, 1)$ (Claim 4.4.1), $\beta_i \in [0, 1]$ (Claim 4.4.6), we can say $(w_i \geq 0 \Rightarrow u_i \geq 0)$ $\forall \mathcal{A}$, irrespective of them reporting truthfully. For truthful reporting in Nash equilibrium, FaRM guarantees strictly positive payoffs which can be easily verified. Hence, FaRM guarantees non-negative utilities for all agents, however, an agent must report truthfully to maximize her utility and maintain her consistency score. This motivates agents to report truthfully but at the same time protects them from loss in case of reporting error.
Furthermore,

$$\Phi(r_i) \in [1, |\mathcal{A}|] \Rightarrow u_i^{\Phi} \in [1, |\mathcal{A}|]$$
$$\alpha_i \in [0, 1) \Rightarrow u_i^{\alpha} \in [0, 1)$$
$$\beta_i \in [0, 1] \Rightarrow u_i^{\beta} \in [0, 1]$$

From the above three equations, $max_{r_i}(w_i) < |\mathcal{A}|$. Consequently,

$$max_{r_i}(u_i) < \frac{B}{|\mathcal{A}|}$$

And hence, in any case scenario it is not possible for the total reward of $|\mathcal{A}|$ agents to exceed budget B. Thus, FaRM is weak budget balanced.

□

Proposition 4.6.2. *FaRM admits selective fairness and cumulative fairness and hence is a fair mechanism.*

Proof. From Claim 4.3.4, we show FaRM admits selective fairness for immediate query. Also, from Claim 4.4.4, we show that FaRM ensures agents are accounted for their consistency and valuable reports. Hence, we conclude FaRM to be a fair mechanism. □

4.7 Summary

FaRM focuses on location-specific queries of people in their everyday life. It can be challenging to obtain prior distribution models for the signal space and hence, prior-free mechanisms are preferable for these set of queries. In spontaneous localized settings all the agents are assumed to observe the same signal. FaRM leverages this property to provide fair rewards to all agents. FaRM's payment scheme consists of three sub-utility structures, namely *report strength*, *consistency score* and *reliability score*. Report strength ensures *selective fairness* for the immediate query by evaluating all agents equivalently against the same result. Consistency score ensures *cumulative fairness* by accounting for an agent's consistency in the system. The report of an agent consistently reporting truthful reports to the system is more valuable than the report of an agent with a lousy reporting history. Consistency score takes the value of a report into consideration. Therefore, it helps in preventing agents from free-riding as it can affect their consistency score negatively. Reliability score estimates an agent's collusion with the nearby agents and reflects the same in the final reward. It motivates agents to earn a higher reward by not colluding with nearby agents and instead stick with their observed signal. Together these scores ensure fair reward to the agents, not just for their immediate contribution but also for their consistent and reliable service to the system. In summary, FaRM is a Nash incentive compatible mechanism which rewards agents fairly for their services.

Chapter 5

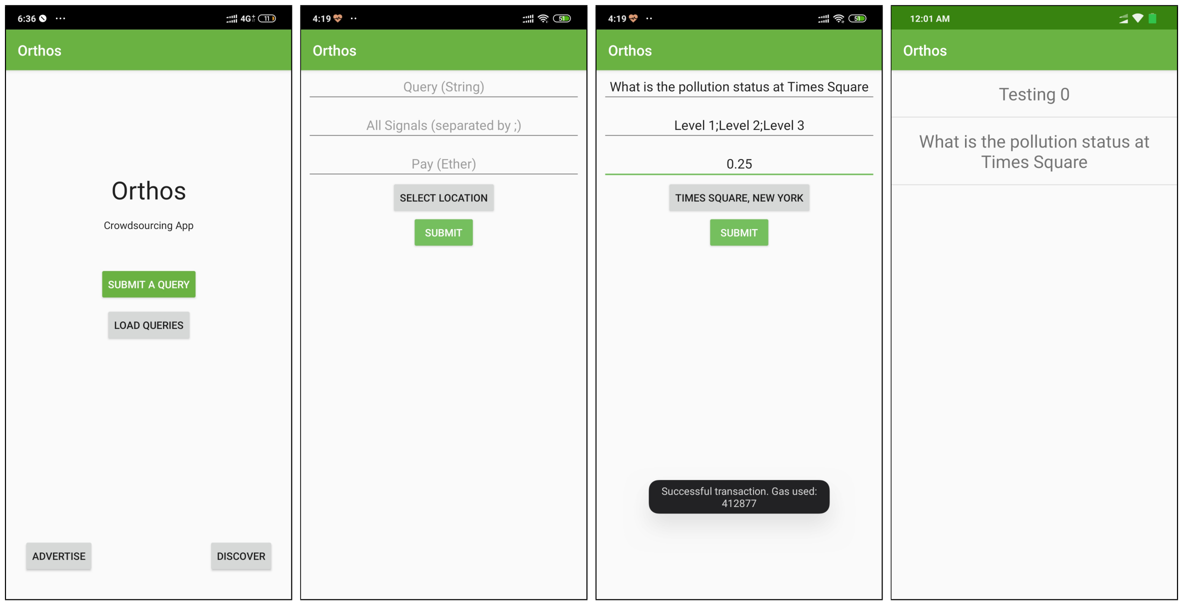
Secure and Trustworthy Crowdsensings over Blockchains in Spontaneous Localized Settings

To ensure agents' truthful participation, crowdsensing mechanisms must guarantee non-negative utilities to agents and employ incentive mechanisms to motivate them to submit accurate reports. Rational agents are expected to maximize their utility while not sacrificing a substantial amount of their resources. Depending on the design of the incentive mechanism, the agents can be rewarded according to the significance of their reports. The existing literature consists of many crowdsensing mechanisms that induce agents to submit truthful reports [11, 15, 21, 22, 33, 34, 35]. We examine state-of-the-art crowdsensing mechanisms and present the necessary conditions for them to be applicable in spontaneous localized settings. After comparing state-of-the-art mechanisms, we argue that the most applicable to the examined settings is the robust peer truth serum for crowdsourcing (RPTSC) [34].

Recent advances in blockchain-based architectures advance the design of decentralized incentive mechanisms. Such architectures are maintained by a network of peers, and motivate agents to participate in crowdsensing applications since their reports will not be controlled by centralized entities. Architectures like Ethereum, support the development of applications that are executed atop blockchain [5] based on *smart contracts*. We use Ethereum smart contracts to develop *Orthos*[1], a trustworthy framework for crowdsensing in spontaneous localized settings. Orthos, via a set of smart contracts, *(i)* processes the submitted reports, *(ii)* estimates the ground truth using weighted averaging techniques and *(iii)* calculates the payments of the agents.

Additionally, Orthos, via developed cryptographic techniques, hides agents' responses to guarantee that agents will not deviate from their honest behavior. Figure 5.1 depicts some activities of the implementation of Orthos on Android. Anyone can download the application of Orthos, submit queries or load queries that request for spatio-temporal information at their location. Every query is defined by *(i)* a String (e.g., How is the availability in restaurant XYZ?), *(ii)* a set of possible answers, *(iii)* the GPS coordinates close to which the responded agents should be when answering the query, and *(iv)* the amount, in gas, the requester is willing to pay. The screenshots of the activities that allow agents to submit their

[1]Orthos, according to Greek mythology, was a two-headed guard dog and the brother of Cerberus. In modern Greek, Orthos means correct and accurate.



What is the pollution status at Times Square
Level 1;Level 2;Level 3
0.25

Successful transaction. Gas used: 412877

Testing 0

What is the pollution status at Times Square

(a) Start. (b) Submit a question. (c) Submit a query (d) See existing queries.

Figure 5.1: Main activities of Orthos.
Anyone can download the application and submit a query or load existing queries in their vicinity. The Orthos mobile application is connected to a set of Ethereum smart contracts.

answers to queries are presented after the description of Orthos in Section 5.3. Following are the main highlights of this chapter[2]:

1) We develop *Orthos* for the development of incentive mechanisms for applications and services that elicit information. It acts as a wrapper for crowdsensing mechanisms and facilitates the collection of agents' reports and the distribution of rewards in a decentralized and privacy-preserving fashion.

2) We design *Proof-of-Location (PoL)* protocol to detect and prevent malicious agents from faking their location. PoL is executed in the mobile devices of the agents to robustly verify that each interested agent can participate if she is located in the correct location.

3) We examine the applicability of Orthos by testing it with a set of 27 participants.

In summary, Orthos works in the trinity of game theory for incentives, mobile computing for location validity, and cryptography and blockchain technology for anonymity.

[2]The preliminary version of this chapter was accepted and presented at EMAS'20[26]

5.1 Problem Formulation

Considering an entity in question EiQ, a set of nearby mobile agents $\mathcal{U}$, and a budget B, we want to estimate a function f (e.g., EiQ can be the Eiffel Tower, f the current queue length in the tickets counter and B can be 1\$). $\mathcal{A}$ ($\mathcal{A} \subseteq \mathcal{U}$) agents choose to participate and assess EiQ. Every agent $i \in \mathcal{A}$ observes a signal $s_i \in S$ and reports a signal $r_i \in S$ which can be different from s_i. After submitting r_i, agent i collects a reward u_i ($\sum_{i \in \mathcal{A}} u_i \leq B$). If the equality holds and the budget is fully utilized, the mechanism is called *Strong Budget Balanced*. Orthos ensures this property while distributing rewards.

5.2 Implementing Decentralized Crowdsensing Mechanisms

Crowdsensing mechanisms can be integrated into decentralized applications (DApps) in the form of smart contracts. Ethereum smart contracts are compiled into bytecode and executed on Ethereum Virtual Machine (EVM). For each computation, the EVM consumes some fuel, named *gas*. Gas is the unit of measurement for the resources consumed in Ethereum. The monetary expenditure depends on the consumed gas units and the gas price at that moment. The gas price is the valuation of gas units in terms of ether and it changes according to market dynamics.

Reading information from a contract is gas-free and nearly instant, however, writing into a smart contract requires gas proportional to the storage needs. Similarly, computations on a smart contract require gas proportional to the computational complexity. Transactions on Ethereum are executed in batches and stored in *blocks*. Each block has a *gas limit* that forces the sum of all the gas needs of the transactions stored on each block to not exceed this limit. Hence, it is not possible to accomplish complex tasks on smart contracts via a single transaction. Also, since storage on the blockchain is expensive, its impractical to maintain long logs of persistent data for a complex task to be carried out in disjoint transactions. It is also worth mentioning that Ethereum does not support floating-point numbers (i.e., all divisions are integer divisions) making computations that require floating-point numbers to be handled on a case by case basis that usually imposes additional computation overhead.

The two primary tasks of any crowdsensing mechanism are collecting and storing reports from all agents and performing computations on those reports to determine rewards for the agents. Both of these tasks are anti-complimentary to the smart contract. In the previous sections, we discussed various crowdsensing mechanisms and presented four that apply to spontaneous localized settings. However, among them, only M16 (PTSC) and M17 (RPTSC) are computationally feasible to implement on the smart contract. M13 is a very complex mechanism with dependency on multiple tasks while M15 uses a logarithm scoring rule which is difficult to implement on the smart contract because of no support for floating-point numbers. PTSC and RPTSC are very similar mechanisms but between them, RPTSC is a more robust mechanism as it excludes the possibility of ill-defined results from PTSC. Hence, we recommend using RPTSC for crowdsensing on decentralized mechanisms. According to our measurements, the gas needs of RPTSC is 2495101, which corresponds to less than half USD.

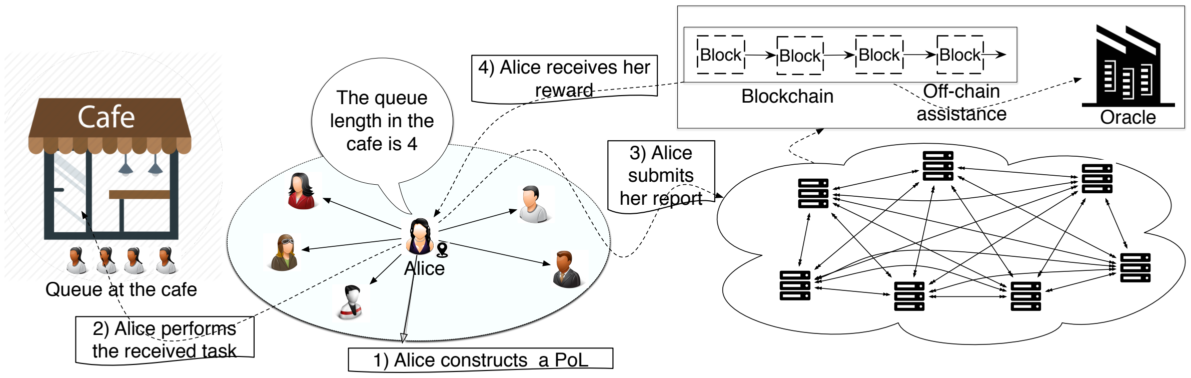

Cafe
Queue at the cafe
The queue length in the cafe is 4
Alice
4) Alice receives her reward
Block
Block
Block
Block
Blockchain
Off-chain assistance
Oracle
3) Alice submits her report
2) Alice performs the received task
1) Alice constructs a PoL

Figure 5.2: Orhtos Overview

5.3 Orthos

Orthos is a blockchain-based crowdsensing mechanism applicable in spontaneous localized settings. RPTSC and any other mechanism that meets the essential criteria presented in Table 3.1 can be applied for crowdsensing in spontaneous localized settings securely and anonymously using Orthos. The architecture of Orthos is split into two parts: a mobile application and a DApp. We have designed a protocol, called *Orthos protocol*, to dictate the interaction between the two components during the crowdsensing process. In Figure 5.2, we show an overview of the Orthos protocol motivated by a basic example, where, agent Alice reports the queue length for a local cafe. Figures 5.1 and 5.4 show a total of six screenshots of the developed mobile application that allows mobile agents to submit a query, load existing queries in their location and answer existing queries by submitting a report.

The Orthos protocol is composed of four phases: *commitment phase*, *reveal phase*, *scoring phase*, and *reward distribution phase*. In the commitment phase, each agent i assesses QS, observes signal s_i and commits to a report r_i. Figure 5.4a shows the screen of the mobile application after the submission of the commitment. No more agents are accepted once this phase ends. Only the final commitment of the agent is taken into consideration and is revealed in the reveal phase where the report is processed, as depicted in Figure 5.4b. Participating mobile agents need to transact with the blockchain part of Orthos to submit their commitments and reveal their reports by calling the `submit()` and `reveal()` smart contract functions respectively.

In the scoring phase, each agent i is rewarded based on her report r_i and the payment mechanism. Crowdsensing mechanisms for spatio-temporal queries are unable to detect if an agent commits a signal after assessing QS at the required location. Agents can attempt to manipulate their location by faking their GPS reading if it is beneficial. Orthos bypasses this limitation using *Proof-of-Location (PoL)*, a distributed protocol that is executed by the agents. PoLs have been used in the design of location-based cryptocurrencies, where agents are required to be either at a specific location to be rewarded [46] or the agents' interconnectivity affects their rewards [8].

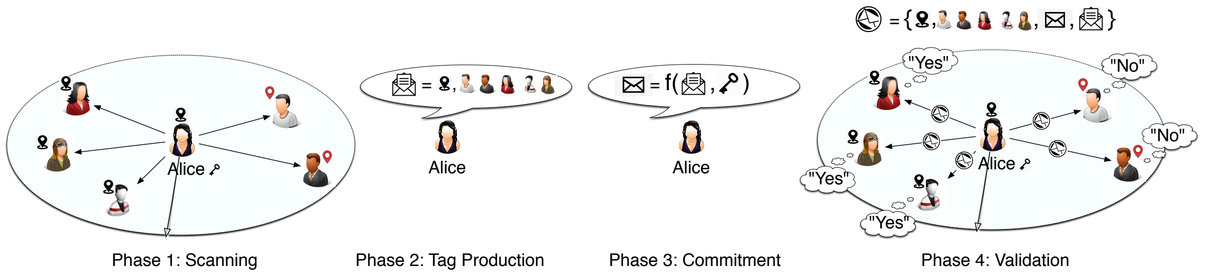


Figure 5.3: Phases of Proof of Location

The four phases a mobile agent follows to produce a proof of her location. Alice, in the first phase, scans her neighborhood and detects five mobile agents. Next, in the second phase, she uses this information and her estimated location to create a tag (opened envelop icon). In the third phase, Alice produces a commitment using the tag and her secret key (closed envelop icon). In the fourth phase, Alice composes a message (envelop in a circle icon) that is composed of the scanned information, the tag, and the commitment and sends it to all her neighbors. Unfortunately, two of her neighbors cannot verify her location since they are not as close to her as needed (red location icons).

5.3.1 Location Proofs in Spontaneous Localized Settings

Agents need to include a PoL whenever they submit an answer to a query for a QS. Using their mobile devices, an agent i that wants to produce a proof of her location broadcasts her *context* to all nearby mobile devices. Orthos is based on Google's Nearby Connections API to connect with mobile phones in Bluetooth and Wi-Fi range. After collecting the broadcasted context, nearby peers respond with their respective contexts. Similar to agent i, all nearby peers exchange their contexts to form their own list of contexts. Then agent i shares this list of contexts with his nearby peers who compare agent i's list of context with their own list of contexts to assess the validity of agent i's location. If valid, each peer responds with a digital signature certifying the validity of agent i's location. Agent i must have enough peer validations to cross the security threshold set by Orthos smart contract. In detail, agent i proves her location is as measured by her GPS or any localization method [4, 6], by following the four phases of the following cryptographic protocol:

Scanning Phase: Scan for neighbours and produce $L_i = \{l_i, \mathcal{P}_i\}$, a message composed of the agent's estimated location, l_i, and her neighbors, $\mathcal{P}_i$.

Tag Production Phase: Use L_i to produce a tag $L_i^T = f(L_i(t))$ of fixed size via a pseudo-random function [20] known to every agent.

Commitment Phase: Use the secret key k_i of agent i to produce a commitment for every neighbor N_i:

$$\mathcal{C}omm t\left(L_i, L_i^T\right) \rightarrow M_i. \tag{5.1}$$

Verification Phase: Every neighbour receives N_i and examines whether user i is at l_i:

$$\mathcal{V}erify\left(L_i, L_i^T, M_i\right) \rightarrow L_{ji}^V \in \{\texttt{yes}, \texttt{no}\}. \tag{5.2}$$

L_{ji}^V equals to "yes" if user j verifies that user i is her neighbour and "no" otherwise. User j returns "yes" if her estimated location has a difference of less than a threshold from the location of agent i.

Every user, after receiving N_i uses the public key of i to extract her location, neighbours and L_i^T. User i, by sending N_i instead of L_i makes sure that her neighbors can only answer to her claim. Misbehaving agents cannot change the location agent i claims to be in. Practically, a malicious agent can only try to produce a PoL for a location she is not currently in. By doing that, she will not be able to verify her fake location by normal agents. Via this process, user i constructs a PoL that a set of her neighbors are within a given distance:

$$\pi_i(QS)\left(M_i, \bigcup_{j\in\mathcal{P}_i} L_{ji}^V, \frac{1}{|\mathcal{P}_i|}\sum_{j=1}^{|\mathcal{P}_i|} 1_{\{L_{ji}^V)==\text{“yes”}\}}\right) \tag{5.3}$$

PoL is defined as the set of messages from the neighbouring devices of a user that the user is at a specific location. Each message is signed by the neighbouring users. Figure 5.3 depicts the four phases of the Proof-of-Location protocol. The Orthos smart contract contains a method named `verifyLocation()` that is responsible for verifying the submitted PoLs from the mobile agents.

5.3.2 Orthos Protocol

A requester can add her query on the network using the `addQuery()` method by specifying the exact query (Q), query location (L), signal space (S), and budget (B). The requester does not need to provide personal information on the network. Once the query is added to the contract, all agents can access it. Next, we present the protocol through which agents can submit and receive a reward for their contributions. For ease of understanding, we consider an arbitrary agent i to walk through the various phases of the protocol.

Commitment Phase: Agents can access all queries of the smart contract and chose to participate in the queries related to a nearby location. Agents can submit their reports using the `submit()` method of the smart contract. For an agent i, `submit()` takes a cipher c_i, which is the commitment ($c_i = keccak256(r_i, k_i)$, where r_i is the reported signal and k_i is the secret key of the agent) of the reported signal, list of peers (identified by their Ethereum addresses) and a list of digital signatures by the peers validating the agent i's location. Every agent is allowed to update her report as long as the phase continues but only the latest report will be considered.

Reveal Phase: Agent i, reveals her commitment by submitting r_i and k_i using the `reveal()` method. The agent report is accepted only if her commitment matches with the reported value i.e. $c_i = keccak256(r_i, k_i)$ and if the submitted proof of location is accepted by the `verifyLocation()` method that implements the verification phase of the PoL protocol.

Scoring Phase: Once the reveal phase is over, agent i calculates the score of her contribution by calling `calcScore()`. Agent i is scored using the requester specified mechanism. When RPTSC is employed, R_i is calculated using Equation 3.5 and the final score is based on R_i, as described by Equation 3.8. The score of agent i is stored on the smart contract before being normalized when all agents have been scored. Agent i gets his reward in the next phase.

Reward Distribution Phase: Once the scoring is finished, agent i adds the corresponding reward to her balance by calling `updateBalance()`. To ensure budget balancing, Orthos normalizes the scores irrespective of the payment scheme and calculates the reward for each agent i by:

$$u_i = \frac{score_i}{\sum_{j \in \mathcal{A}} score_j} \times B, \tag{5.4}$$

where B is the total budget for the request. Agents can call `getBalance()` to get their balance and `withdraw()` to transfer it to their Ethereum accounts.

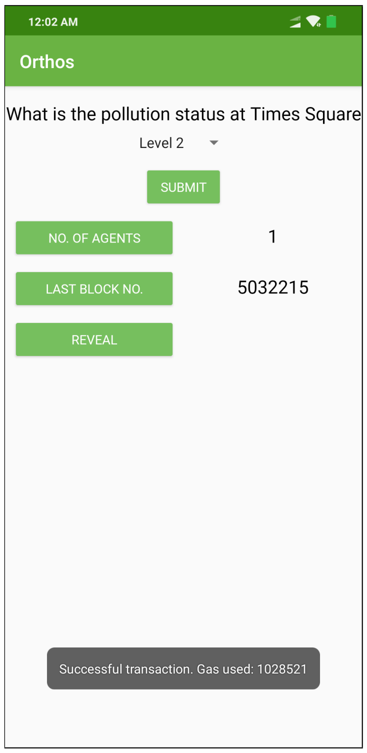
5.4 Performance Evaluation

We implement Orthos as a decentralized application (DApp) that is composed of Ethereum smart contracts that are deployed on Rinkeby Testnet Network[3] and an Android mobile application that is presented in Figures 5.1 and 5.4. We measure the gas needs and the cost in USD of each implemented method of Orthos. Additionally, we recruited 27 students (18 male and 9 female) with an average age of 22 years and asked them to install Orthos and participate as mobile agents. We generated a query to ask them about the difficulty of the subject and provided four signals. We used the acquired data to measure gas costs for executing Orthos.

5.4.1 Implementation Details

Agents are identified on Ethereum using their Ethereum address. For every new account, Ethereum generates a random pair of a public key and the correspondent private key. The keys are completely unrelated to the real-world identity of the agent, hence, granting an anonymous medium of participation to the agent. The Ethereum address is the last 20 bytes of the hash of the public key. Agents are encouraged to create a new Ethereum address for every new query to avoid any privacy leaks. The mobile part of Orthos is built to target mobile devices with Android SDK version 28 and supports devices with minimum SDK version 23. To connect a mobile device with the Ethereum blockchain, the device must host an Ethereum node. However, the hosting of an Ethereum node on a mobile device is energy demanding and demotivating for mobile agents. As a solution to this problem, we use the Infura API[4]. Infura is a hosted Ethereum node cluster that supports JSON-RPC over HTTPS and WebSocket interfaces

[3]rinkeby.etherscan.io
[4]https://infura.io/docs


12:02 AM

Orthos

What is the pollution status at Times Square

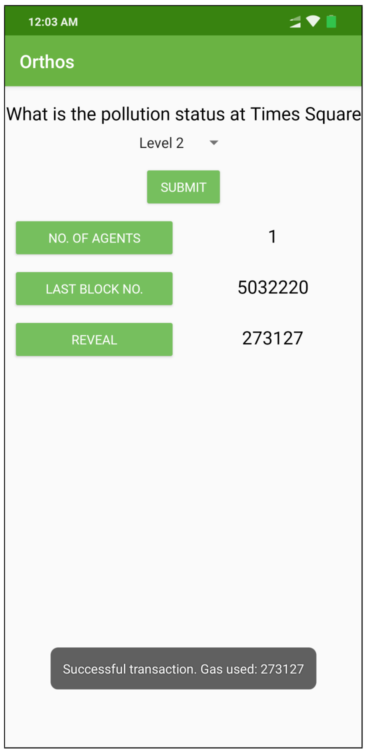
Level 2

1

5032215

Successful transaction. Gas used: 1028521

(a) Commit query answer.


12:03 AM

Orthos

What is the pollution status at Times Square

Level 2

1

5032220

273127

Successful transaction. Gas used: 273127

(b) Reveal query answer.

Figure 5.4: Submitting a report

After submitting an answer, by triggering the `submit()` method, the developed application waits until the deadline to reveal the submitted answer, by calling the `reveal()` method. After that, Orthos calculates the score of each report and distributes the rewards.

and allows mobile agents to perform requests and set up subscription-based connections to Ethereum blockchains. Once the connection to the Ethereum blockchain is established, we integrate wrapper functions to the mobile part of Orthos to automate the call of smart contract methods on the blockchains. We use *Web3j* to generate equivalent wrapper functions of the smart contract for Java/Kotlin which we then use for the development of the mobile part.

5.4.2 Experiments

Orthos enables mobile agents to both add queries and respond to existing ones. Table 5.1 lists the methods that require gas to be executed while Table 5.2 lists additional helping methods for secondary functionalities such as payments between the query requester and the responding mobile agents. Table 5.1 shows that the required gas for deploying a smart contract that implements RPTSC (`ContractCreation`) is 2495101, which corresponds to less than half USD. Adding a new query for spatio-temporal data using `addQuery()` requires only 8 cents. Every participating mobile agent spends 19 USD cents for the call of `submit()` on the first phase of Orthos protocol, 6 USD cents during the second phase for

Name	**Gas Used**	**USD Cost**
`ContractCreation()`	2495101	0.47
`addQuery()`	442183	0.08
`submit()`	1013457	0.19
`reveal()`	74138	0.01
`verifyLocation()`	183733	0.03
`calcScore()`	6431116	1.20

Table 5.1: Gas Consumption Table

Gas consumption for deploying Orthos, and transact with it on Rinkeby Ethereum testnet. For converting gas to USD we used the default gas price (1 GWei) and the price of Ethereum on 13-Nov-2019. (i.e., USD = gas$\cdot 188 \cdot 10^{-9}$).

Name	**Description**
`getScore()`	Returns the score for a particular query
`getBalance()`	Returns the total balance of an agent in the protocol
`withdraw()`	Withdraws the total balance from the protocol
`updateBalance()`	Updates agents' balance after the query

Table 5.2: Helping Methods

`getScore()` and `getBalance()` are gas-free while the gas needs of `withdraw()` and `updateBalance()` are negligible.

the calls of `reveal()` and `verifyLocation()`, and 1.2 USD for the calculation of her score via the `calcScore()` on the third phase. The collection of the reward is gas-free. Note that for the cost calculation we considered the default gas price that leads to the completion of each call within 15 seconds. Lower gas prices can reduce the cost for each mobile agent but delay the collection of the data. Depending on the deadline of a query, the mobile agents are responsible to device the gas price they are willing to use for submitting their readings.

5.5 Design Tradeoffs

We design Orthos as an Ethereum-based framework that functions via smart contracts. Although the use of smart contracts on every component of Orthos guarantees its auditability and generalisability, it increases the cost of its operation in terms of gas. In this section we discuss the design tradeoffs for the stored data and the computational demanding components of Orthos.

Storage	Gas	Cost (ETH)	Cost(USD)
256-bit word	20000	0.00002	0.177
1 MB (31250 words)	625×10^6	18.75	5531.25
1 GB (1000 MB)	625×10^9	18750	531250

Table 5.3: Contract Storage Costs
As of 13-Nov-2019, 1 ETH = 188 USD and 1 gas = 10^{-9} ETH.

5.5.1 Storage requirements

Actions performed on Orthos are recorded as transactions and get logged to the Ethereum blockchain. Anyone can access these logs and verify the operations of Orthos. There are two methods to store persistent data on the smart contract, *contract storage* and *log storage*. Data stored on the *contract storage* can be accessed by the corresponding smart contract and other smart contracts depending on the permissions provided. However, the cost of storing data on the *contract storage* is very high, and therefore only state variables and only the most crucial data required by the smart contract should be stored there. Table 5.3 provides cost details for *contract storage*. Orthos stores agent commitments and their reports on the *contract storage* as it needs it to verify agent reports and then use it to compute their scores. The contract also stores the agents' PoLs which are required to validate their location.

A cheaper alternative is *log storage* where data is stored on transaction logs created by triggering events[5]. For every log event, the gas price is:

$$Gas\_prince = 8 * (nBytes) + 375 * (1 + iArgs),$$

where $nBytes$ denotes the number of bytes and $iArgs$ the number of the indexed arguments. A limitation to this form of storage is that smart contracts cannot access directly the data stored on *log storage* and need additional functions for that. Another alternative is to use external storage (e.g., IPFS [2]) and store hashed of the externally stored data. Unfortunately, this increases the required setup on the agents' mobile devices.

5.5.2 Reward Calculation

It is possible to store the reports on the logs and then use third party services (*oracles*) to compute the agent reward using the logged data. In this way, each agent will save more than 50% of her participation cost. However, the requester will have to cover the fees for the oracle service provider. *Provable*[6] is one such popular oracle service provider designed to act as an untrusted intermediary. Provable is referred to as a *provable honest* service as it provides cryptographic proofs showing that the data they provide is

[5]solidity.readthedocs.io/en/v0.4.24/contracts.html#events
[6]https://provable.xyz

		Proof Type		
Data Source	**Base Price**	TLSNotary	Android	Ledger
URL	0.01$	+0.04$	+0.04$	N/A
WolframAlpha	0.03$	N/A	N/A	N/A
IPFS	0.01$	N/A	N/A	N/A
random	0.05$	N/A	N/A	+0.0$
computation	0.50$	+0.04$	+0.04$	N/A

Table 5.4: Provable Fee Structure.

really the one that the server gave them at a specific time. It works in the following way: first, a smart contract uses the provable API to request for a task execution off-chain, and then *Provable* performs the task off-chain and makes a callback transaction to provide the results of the task and the proof of authenticity. With each request, the contract must pay enough fee to *Provable* to execute the task and send a callback transaction. The fee consists of two parts: The gas that corresponds, using a recent exchange rate, to the USD price for the data source and the authenticity proof requested and the gas *Provable* will spend for sending the callback transaction. Table 5.4 provides fee details for the data source and authenticity proof.

Note that using an oracle service to compute the rewards off-chain defeats the purpose of an otherwise decentralized framework as the services centrally compute all the rewards. Hence, the rewards are computed on blockchain and stored on public storage to maintain transparency. The rewards are only associated with Ethereum addresses and therefore do not compromise privacy even though stored publicly. We allow agents to accumulate their reward as a balance on the smart contract and retrieve it whenever they want. It is a design trade-off to avoid repetitive transactions that pay agents agent for every query.

5.6 Summary

Smart cities require constant and accurate data to function properly. Existing crowdsensing systems are built on centralized architectures that imply trusting third parties on providing reliable and secure services. Motivated by these challenges in robust spatio-temporal information acquisition in smart cities, we proposed *Orthos*, a blockchain-based framework that enables the deployment of crowdsensing mechanisms. After introducing the necessary characteristics of crowdsensing mechanisms in spontaneous localized settings and analyzing the state of the art, we concluded that RPTSC is the most suitable. Additionally, we proposed the Proof-of-Location protocol to assist Orthos on guaranteeing that agents participating in crowdsensing mechanisms are at the expected locations when reporting their measurements. We used Ethereum smart contracts to develop the methods needed to support any crowdsensing mechanism. To test Orthos and assess its applicability, we deployed its smart contracts on a popular Ethereum testnet and developed an Android Application to perform experiments with a live audience. In summary, Orthos assist agents in posting their queries and answer others' queries on their mobile devices at extremely very low rates. It protects agents' privacy and provides a secure and transparent platform for exchange and acquisition of information with no tampering or interference by any centralized entity.

Chapter 6

Conclusions and Directions for Future Work

Existing crowdsensing mechanisms are focused on settings that are relevant to mass population, e.g., online reviews about products and services, and community sensing regarding prevalent societal topics (pollution, global warming, etc.). In this work, we focused on location-specific queries that may be relevant to any general crowd. These queries are spontaneous, and hence, there may not exist any prior distribution of signals for these queries. We formally defined the settings (called *spontaneous localized settings*) for these queries and identified the necessary characteristics any mechanism must satisfy to be applicable for them. We evaluated the applicability of 17 crowdsensing mechansims in our settings based on those characteristics. Most notably, we identified two primary challenges to mobile crowdsensing in our settings, i) fairness in reward distribution, and ii) secure and trustworthy framework.

For the first challenge, we introduced two notions of fairness, *selective* and *cumulative*, to motivate fair reward distribution in our crowdsensing settings. We proposed FaRM, a fair reward mechanism for our settings that satisfies both these notions of fairness, incentivizes agents to report truthfully and resists collusion from malicious group of agents.

For the second challenge, we created a framework on a secure and transparent platform i.e., blockchain to ensure all transactions and actions are auditable and accountable. We implemented the framework on Ethereum blockchain and provided the gas statistics required for executing crowdsensing mechanism on smart contract. With this, we enabled existing crowdsensing mechanisms to operate on a secure and trustworthy platform.

6.1 Future Work: Integrating FaRM with Orthos

FaRM is a tailor-made mechanism for spontaneous localized settings. It ensures selective and cumulative fairness in reward distribution. Orthos framework is designed for secure and trustworthy crowdsensing in spontaneous localized settings. Together, they form secure, trustworthy, and fair crowdsensing system.

They limitation of FaRM is the computational complexity of scores. It is expensive to be performed on a blockchain based framework such as Orthos. Particularly *reliability score* has total $O(n^2)$ complexity for all agents. In our future work, we aim to formulate a heirarchial mechanism that takes advantage of distributed programming to reduce the computational complexity of the scores, and thus present a complete system (mechanism + framework) that is secure, trustworthy and fair. We also leave it for the extension to look for other blockchain platforms for Orthos.

Related Publications

[1] M. H. Moti, D. Chatzopoulos, P. Hui, and S. Gujar. FaRM: Fair Reward Mechanism for Information Aggregation in Spontaneous Localized Settings. *In Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI 2019.

[2] M. H. Moti, D. Chatzopoulos, P. Hui, B. Faltings, and S. Gujar. Orthos: A Trustworthy AI Framework For Data Acquisition. *In Proceedings of the 8th International Workshop on Engineering Multi-Agent Systems*, EMAS@AAMAS 2020.

Other Publications

[1] S. Damle, M. H. Moti, P. Chandra and S. Gujar. Aggregating Citizen Preferences for Public Projects Through Civic Crowdfunding. *In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS 2019.

[2] S. Damle, M. H. Moti, P. Chandra and S. Gujar. Civic Crowdfunding for Agents with Negative Valuations and Agents with Asymmetric Beliefs. *In Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI 2019.

[3] S. Damle, M. H. Moti, P. Chandra and S. Gujar. Designing Refund Bonus Schemes for Provision Point Mechanism in Civic Crowdfunding. *In Proceedings of the 2nd Games, Agents, and Incentives Workshop*, GAIW@AAMAS 2020.

Bibliography

[1] M. Abraham and K. Jevitha. Runtime verification and vulnerability testing of smart contracts. In *International Conference on Advances in Computing and Data Sciences*, pages 333–342. Springer, Springer, 2019.

[2] J. Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[3] A. Bogner, M. Chanson, and A. Meeuw. A decentralised sharing app running a smart contract on the ethereum blockchain. In *Proceedings of the 6th International Conference on the Internet of Things*, pages 177–178. ACM, ACM, 2016.

[4] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE personal communications*, 7(5):28–34, 2000.

[5] V. Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[6] S. Čapkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad hoc networks. *Cluster Computing*, 5(2):157–167, Apr 2002.

[7] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui. Localcoin: An ad-hoc payment scheme for areas with high connectivity: Poster. In *MobiHoc '16*, pages 365–366. ACM, 2016.

[8] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui. Localcoin: An ad-hoc payment scheme for areas with high connectivity. *CoRR*, abs/1708.08086, 2017.

[9] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui. Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain. In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 442–450. IEEE Computer Society, Oct 2018.

[10] ConsenSys. Mythril. https://github.com/ConsenSys/mythril, 2017.

[11] A. Dasgupta and A. Ghosh. Crowdsourced judgement elicitation with endogenous proficiency. In *Proceedings of the 22nd international conference on World Wide Web*, pages 319–330, 2013.

[12] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[13] J. Eberhardt and S. Tai. On or off the blockchain? insights on off-chaining computation and data. In *European Conference on Service-Oriented and Cloud Computing*, pages 3–15. Springer, 2017.

[14] Ethereum. Remix. https://github.com/ethereum/remix, 2016.

[15] B. Faltings, J. J. Li, and R. Jurca. Incentive mechanisms for community sensing. *IEEE Transactions on Computers*, 63(1):115–128, 2014.

[16] J. Feist, G. Grieco, and A. Groce. Slither: a static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 8–15. IEEE, IEEE / ACM, 2019.

[17] N. Goel and B. Faltings. Deep bayesian trust : A dominant and fair incentive mechanism for crowd. In *To appear in Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

[18] G. Han, L. Liu, S. Chan, R. Yu, and Y. Yang. Hysense: A hybrid mobile crowdsensing framework for sensing opportunities compensation under dynamic coverage constraint. *IEEE Communications Magazine*, 55(3):93–99, 2017.

[19] S.-W. Huang and W.-T. Fu. Systematic analysis of output agreement games: Effects of gaming environment, social interaction, and feedback. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[20] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 12–24. ACM, ACM, 1989.

[21] R. Jurca and B. Faltings. An incentive compatible reputation mechanism. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pages 285–292. IEEE, 2003.

[22] R. Jurca and B. Faltings. Robust incentive-compatible feedback payments. In *Agent-Mediated Electronic Commerce. Automated Negotiation and Strategy Design for Electronic Markets*, pages 204–218. Springer, 2007.

[23] N. Lambert and Y. Shoham. Truthful surveys. In *International Workshop on Internet and Network Economics*, pages 154–165. Springer, 2008.

[24] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269. ACM, ACM, 2016.

[25] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359–1373, 2005.

[26] M. H. Moti, D. Chatzopoulos, P. Hui, B. Faltings, and S. Gujar. Orthos: A trustworthy ai framework for data acquisition. In M. H. Moti, editor, *8th International Workshop on Engineering Multi-Agent Systems (EMAS 2020)*, 2020.

[27] M. H. Moti, D. Chatzopoulos, P. Hui, and S. Gujar. Farm: Fair reward mechanism for information aggregation in spontaneous localized settings. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 506–512. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[28] Y. Narahari. *Game theory and mechanism design*, volume 4. World Scientific, 2014.

[29] D. Prelec. A bayesian truth serum for subjective data. *science*, 306(5695):462–466, 2004.

[30] M.-R. Ra, B. Liu, T. L. Porta, and R. Govindan. Medusa: A Programming Framework for Crowd-Sensing Applications. In *MobiSys*. ACM, June 2012.

[31] G. Radanovic and B. Faltings. A robust bayesian truth serum for non-binary signals. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI" 13)*, pages 833–839, 2013.

[32] G. Radanovic and B. Faltings. Incentives for truthful information elicitation of continuous signals. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 770–776, 2014.

[33] G. Radanovic and B. Faltings. Incentive schemes for participatory sensing. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1081–1089. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[34] G. Radanovic, B. Faltings, and R. Jurca. Incentives for effort in crowdsourcing using the peer truth serum. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(4):48, 2016.

[35] B. Riley. Minimum truth serums with optional predictions. In *Proceedings of the 4th Workshop on Social Computing and User Generated Content (SC14)*, 2014.

[36] S. Robertson, M. Vojnovic, and I. Weber. Rethinking the esp game. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 3937–3942. ACM, 2009.

[37] N. Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[38] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 9–16. IEEE, ACM, 2018.

[39] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev. Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 67–82. ACM, ACM, 2018.

[40] L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

[41] B. Waggoner and Y. Chen. Output agreement mechanisms and common knowledge. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[42] L. Wang, D. Zhang, Z. Yan, H. Xiong, and B. Xie. effsense: A novel mobile crowd-sensing framework for energy-efficient and cost-effective data uploading. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(12):1549–1563, Dec 2015.

[43] J. Witkowski and D. C. Parkes. Peer prediction with private beliefs. In *Proceedings of the 1st Workshop on Social Computing and User Generated Content (SC 2011)*, 2011.

[44] J. Witkowski and D. C. Parkes. Peer prediction without a common prior. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 964–981. ACM, 2012.

[45] J. Witkowski and D. C. Parkes. A robust bayesian truth serum for small populations. In *AAAI*, volume 12, pages 1492–1498, 2012.

[46] L. Wolberger, A. Mason, and S. Capkun. Platin - proof of location protocol on the blockchain. https://platin.io/, 2018.

[47] R. Xu, Y. Chen, E. Blasch, and G. Chen. Blendcac: A smart contract enabled decentralized capability-based access control mechanism for the iot. *Computers*, 7(3):39, 2018.

[48] P. Zhang and Y. Chen. Elicitability and knowledge-free elicitation with peer prediction. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 245–252. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[49] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan. Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*, 6(2):1594–1605, 2018.