

RESEARCH ARTICLE

A Context-Aware Blockchain-Based Crowdsourcing Framework: Open Challenges and Opportunities

MAHA KADADHA¹, (Member, IEEE), SHAKTI SINGH^{1,2}, RABEB MIZOUNI^{1,2}, (Member, IEEE), AND HADI OTROK^{1,2}, (Senior Member, IEEE)

¹EECS Department, Khalifa University, Abu Dhabi, UAE

²Center for Cyber-Physical Systems, Khalifa University of Science and Technology, Abu Dhabi, UAE

Corresponding author: Maha Kadadha (maha.kadadha@ku.ac.ae)

This work was supported by the Khalifa University of Science and Technology-Competitive Internal Research Award under Grant CIRA-2020-028.

ABSTRACT Crowdsourcing is a rapidly growing paradigm that commercial platforms such as Amazon MTurk and UpWork are adopting for allocating tasks to workers. Such frameworks typically employ a centralized infrastructure to implement required mechanisms such as task allocation, submission evaluation, and payment computation. However, centralized deployment comes with unresolved challenges in terms of trust, reliability, and transparency. Blockchain technology has been embraced for the deployment of crowdsourcing frameworks to enable trusted and autonomous execution. Each of the existing Blockchain-based crowdsourcing/ crowdsensing framework targets a specific application context due to the constraint capabilities of Blockchain. In this paper, we propose a context-aware Blockchain-based crowdsourcing framework where the context is defined by task requirements and workers' availability. The proposed framework is developed upon the review of existing works integrating Blockchain and crowdsourcing where the challenges and future directions are identified. The proposed framework has two classes of components: 1) *core components* implementing the basic framework functionalities, and 2) *advanced components* which are context and data managers that help improve the framework performance. The *Advanced Context Manager* is designed to monitor the current context and select the mechanisms to run for the core components accordingly. The core components are implemented as smart contracts on Blockchain for autonomous and trusted execution, while the advanced components are implemented spanning Blockchain and the cloud for flexibility and scalability. A case study demonstrating the performance of context-aware task allocation algorithms is presented. It shows how capturing the current system context can help achieve better overall performance based on the objective of the sensing application under consideration.

INDEX TERMS Crowdsourcing, blockchain, smart contracts, task allocation, context.

I. INTRODUCTION

The increasing connectivity of users has paved the way for the wide deployment of crowdsourcing frameworks. Crowdsourcing [1] is a contemporary practice to answer tasks from end-users (requesters), being service or data requests, by crowd members (workers). It has been used in multiple domains such as environment monitoring [2], [3],

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak¹.

healthcare [4], [5], transportation [6], [7], and social networks [8], [9]. Commercial crowdsourcing applications have recorded high user engagement as for Uber (3.9 million drivers¹), Amazon Mechanical Turk MTurk (100 thousand Turkers), and GigWalk (1.5 million Gigers).

A critical component in crowdsourcing applications is the intermediary platform deployed by an organization or cooperation to manage the interactions between requesters

¹<https://www.uber.com/newsroom/company-info/>

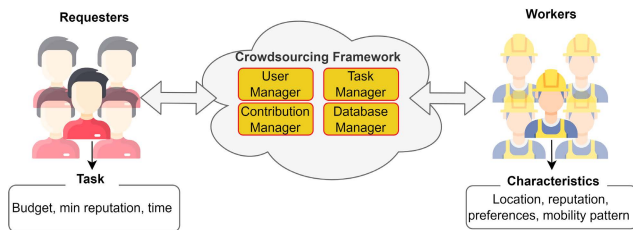


FIGURE 1. Crowdsourcing system model [10].

and workers. This platform is responsible for collecting tasks from requesters, allocating high-performance workers, and collecting their submissions. Furthermore, it evaluates and aggregates workers' submissions before returning them to the requesters for payments.

Fig. 1 illustrates a centralized crowdsourcing framework with some of its core components [10]. Each component hosts computational mechanisms required for its operation. For example, *User Manager* holds functions for user registration while *Task Manager* hosts functions for task creation and allocation. The functions within the managers differ based on the application the framework targets.

Different challenges arise when trying to propose an efficient crowdsourcing platform due to the diverse requirements of requesters and the different capabilities of workers. Requesters create location-based tasks with time constraints and some quality requirements. On the other hand, workers differ in their task eligibility, availability, and reliability. Selecting the appropriate workers is crucial to the framework's success.

In addition, while centralized crowdsourcing frameworks can optimize performance, they are susceptible to security threats in terms of the availability and quality of the provided service. In fact, in April 2015, Uber China users were faced with a service break that prevented them from ending their ride once reaching their destinations [11]. Biased execution arises from the concealed mechanism execution at the framework. An example is MTurk, which does not validate requesters' decisions for fairness, allowing requesters to potentially misbehave.

The existing research works in the crowdsourcing domain primarily focus on - 1) proposing appropriate mechanisms that answer crowdsourcing needs [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], and 2) improving the framework's deployment infrastructure, for instance by integrating Blockchain [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37].

In the literature [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], diverse task allocation algorithms are proposed based on greedy, optimization, auction, and stable matching theory. Each of these algorithms guarantees different properties for allocated tasks and workers. However, the main limitation of these existing works is that they are designed for a specific application area such as environment monitoring, delivery, ride-sharing, etc. In addition, their performance varies based on the context

defined by the number of tasks (demand), the number of workers (supply), and task requirements.

Meanwhile, Blockchain has been applied to different applications such as Crowdsensing [34], [37], VANET [38], Supply chain logistics [39], and News Tracing [40]. In specific, Blockchain-based crowdsourcing frameworks [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37] have been proposed to overcome the limitations of centralized deployment. Blockchain [41] was proposed for trusted exchange of Bitcoin between users and was extended to Ethereum [42] for execution of programs on-chain- smart contracts [43]. Smart contracts are executed in a decentralized, transparent, and autonomous manner without a trusted third party. Commercial blockchain-based crowdsourcing applications are emerging using this paradigm such as Drive ride-sharing application² developed as a replacement for Uber, and Taskopus competing with MTurk [44].

Commercial and research frameworks are migrating some functionalities of the framework such as task allocation, submission evaluation, and worker payment to the Blockchain, making them benefit from its properties, mainly the execution transparency. However, Blockchain-based frameworks require computationally efficient functions to maintain the cost efficiency of the framework. In addition, each framework is designed for a specific application requirement and cannot adapt to the change in context between different applications.

In summary, the research problem this work tries to address is: How to design a crowdsourcing framework that can provide trusted execution while answering different crowdsourcing application contexts including the application domain of the sensing task, its urgency, the current worker's availability in the area of interest?

To the best of our knowledge, this paper presents the first context-aware Blockchain-based crowdsourcing framework. First, the challenges and opportunities in general crowdsourcing and Blockchain-based crowdsourcing frameworks are identified and discussed. Second, a Blockchain-based crowdsourcing framework is proposed, which is capable of answering the requirements of different contexts. The proposed framework consists of core and advanced components that manage the crowdsourcing process. The core components manage the task allocation, the contribution evaluation, the payment, and the user feedback. These components are hosted as smart contracts on Blockchain for autonomous and transparent execution. The advanced components are context and data managers, which are implemented using the resources of Blockchain and/or the cloud for scalability.

The proposed context manager is responsible for monitoring the current application requirements and demand on the framework to intelligently switch the used mechanisms by the framework's core components. Meanwhile, the data manager is responsible for the data relevant to the framework users and created tasks. It is implemented as part of the Blockchain and the cloud to provide alternatives for data storage.

²<https://www.drife.io/>

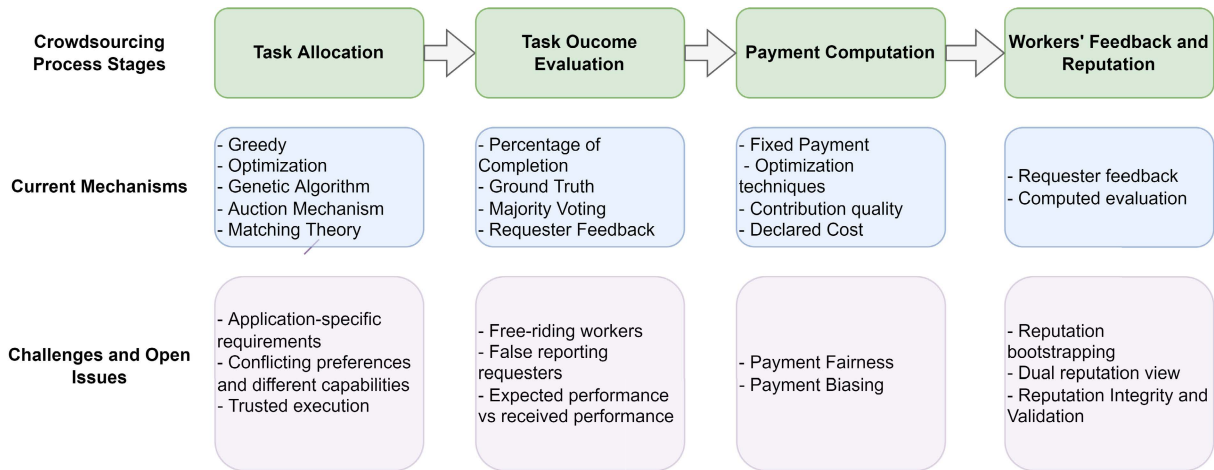


FIGURE 2. Challenges in crowdsourcing framework process.

Data storage on the Blockchain provides transparency and traceability though it comes with a high monetary cost. On the other hand, cloud storage is cost-efficient yet is not tamper-proof. The manager organizes the framework's data between the Blockchain and the cloud. In addition, it analyzes the available data to improve the framework.

A case study focusing on the task allocation component is presented using a real-life dataset to understand the change in the performance of possible task allocation algorithms under different contexts.

Overall, our contributions are summarized as follows:

- A review for general and blockchain-based crowdsourcing frameworks to identify remaining challenges.
- An evaluation for different task allocation mechanisms for blockchain-based crowdsourcing frameworks.
- A context-aware blockchain-based crowdsourcing framework capable of answering different application contexts.

The rest of the paper is organized as follows. Section II highlights the current efforts and open challenges in the areas of crowdsourcing and Blockchain. Section III presents the proposed Blockchain-based crowdsourcing framework. Section IV details the conducted case study. Section V concludes the paper.

II. CURRENT EFFORTS AND OPEN CHALLENGES IN INTEGRATING CROWDSOURCING AND BLOCKCHAIN

In this section, the current efforts in crowdsourcing are summarized. In addition, the efforts targeting Blockchain-based crowdsourcing are summarized to identify open research gaps.

A. CROWDSOURCING

There are four main stages a published crowdsourcing task goes through: 1) workers' selection, 2) task outcome evaluation, 3) payment computation, and 4) workers' feedback and reputation. This section presents a summary for each stage as well as open challenges in centralized crowdsourcing

deployment. Fig. 2 summarizes the used mechanisms and challenges at each of these stages.

1) WORKERS' SELECTION

The first stage a task goes through is its allocation to available worker/s. The allocation of the right task to the right worker is the processor for successful task completion. The allocation process starts with understanding the requirements and constraints of a task as it affects the eligibility of workers. A task's requirements and constraints mainly include the required data type, location, time, quality, and the number of required responses. It is important to consider these requirements in the selection of the task allocation mechanism. Tasks can be allocated either by an allocation mechanism or by volunteer workers. Nevertheless, task allocation is used to maintain the scalability of the framework and reduce the complexity of workers trying to identify suitable tasks when many tasks are concurrently available [45].

The existing works adopt different allocation algorithms based on the requirements of available tasks. The algorithms use greedy algorithms, optimization techniques [12], [13], genetic algorithm [14], [15], [16], auction mechanisms [17], [18], [19], [20], [21], [22], and stable matching theory [23], [24], [25], [26]. Each of the algorithms offers different properties and is selected based on the objective of the framework. For instance, greedy algorithms do not offer an optimal solution but tend to be computationally efficient, while optimization techniques provide an optimal solution but are computationally expensive. Auction mechanisms motivate sharing truthful information and are applicable when workers and/or requesters are required to share their information. Matching theory provides a methodology to pair workers and requesters from two different sets which can lead to stable matches being formed. Therefore, it is beneficial when they are of different preferences.

2) TASK OUTCOME EVALUATION

The evaluation of worker contribution provides constructive feedback about the submitted data or completed task by a

worker in terms of its quality. This assessment is important for the requester and the framework to determine the worker's payment. The worker contribution evaluation differs based on the task type. Multiple approaches have been used such as the completion percentage of the task, group truth, majority voting, or requester feedback. The completion percentage is useful for tasks concerned solely with the completion of the tasks such as surveys and rides. Evaluation against a ground truth is useful for sensing tasks when the outcome is a fixed known value. However, as this is not always available, majority voting becomes helpful as it relies on collecting submissions from multiple workers about the expected result of a task and taking the most occurring result as the correct one. Voting is feasible with a finite number of possible results but is too complicated when the result is from an infinite number of possibilities. On the other hand, direct requester feedback is used for design tasks where the evaluation is subjective to the requester's opinion.

While some of the existing works such as [12], [20], and [23] evaluate workers' contribution, other works rely on the expected performance of allocated workers as an indication for the actual performance of workers.

3) PAYMENT COMPUTATION

Monetary incentive motivates workers to engage in crowdsourcing activity. The payment amount needs to be computed based on the entitlement of the workers and the dedicated budget of the task. Depending on the used approach and the nature of the task, payment per worker could be fixed or computed using an optimization technique based on the contribution quality, or the declared cost. For instance, payments based on the contribution quality are widely adopted in crowdsensing where the quality of the submitted data is evaluated to determine the worker's entitled payment. Alternatively, a combination of a fixed payment and the declared cost is used in ride-sharing tasks to accommodate the worker's interest and the requester's budget.

Auction mechanisms have been used to compute workers' payments based on their declared bids in the works [13], [17], [19], [20], [21], [22]. There are multiple types of auctions, such as first-price auction, second-price auction, and double auctions. Double auctions are used to consider the payment a requester is capable of paying and the payment a worker is interested in acquiring. This auction is truthful, which implies that workers and requesters will maximize their gain by bidding their true valuations for the available tasks. For example, if a worker/ requester bid a value higher or lower than their true valuation for a task, they might either be unallocated or overpay for the task being completed.

4) WORKERS' FEEDBACK AND REPUTATION

The last stage for a task is collecting and compiling feedback about workers and requesters to have an indication about their trust and commitment. One metric that reflects the trust of the users is their reputation, computed based on the historical interactions of the users [48]. The feedback and reputation is

important for better task allocation in consequent tasks. It can be compiled for workers and requesters in the framework, yet the existing centralized works mostly compile the reputation of workers. The worker reputation is accounted for during task allocation, yet it is assumed to be available without proposing a model to compute it. On the other hand, few existing works such as [12], [15], [16], and [23] present a worker reputation computation model within their proposed frameworks.

B. CROWDSOURCING CHALLENGES

There are different challenges and open issues that impact crowdsourcing frameworks at different stages. The challenges can be divided into 1) optimization challenges and 2) User strategy challenges. Each type is elaborated on in the following sections.

1) OPTIMIZATION CHALLENGES

The crowdsourcing process entails optimization at the different stages to be able to allocate tasks to workers and evaluate the worker's outcome. There are multiple aspects that need to be accounted for during the optimization, which makes the optimization problem challenging. Below are a few aspects.

a: TASK-SPECIFIC REQUIREMENTS

The different application areas that crowdsourcing is applied to such as sensing, delivery, and health emergency vary in the requirements for their tasks. The type of the task, location-dependency, and time-dependency are a few of the requirements that affect the allocation scheme applicable in the framework. For example, environment sensing tasks are usually location-dependent and delay-tolerant, unless in the scenario of a natural disaster. They also require the availability of sensors when performing the allocation and demand high-quality data. Ride-sharing tasks are time-sensitive and require the availability of the vehicle for the ride. Emergency health tasks are time-sensitive and location-dependent while being medical image labeling tasks are delay-tolerant and location-independent [46]. Hence, different algorithms are required to answer the diverse requirements of the tasks.

b: DIFFERENT WORKERS' CAPABILITIES

Workers vary in their abilities (computation, sensing, knowledge level) and reputation metrics. It is crucial to account for such information in the allocation as competent workers are more capable of fulfilling tasks with high quality. Workers' capabilities may vary in time depending on the dynamically changing conditions, which consequently affects the allocation.

c: REPUTATION BOOTSTRAPPING

When a worker joins the framework, an initial reputation needs to be assigned to them. While justifiable, the adoption of a low initial reputation might contribute in reducing the worker's eligibility for available tasks, which excludes workers for possible tasks that could increase their reputation.

Alternatively, a high reputation might be too optimistic and grant the worker priority over more competent workers for the task. Different mechanisms have been used to bootstrap reputation values such as initializing a similar reputation to similar agents [49], based on their social network [50], or based on collected endorsements [51].

2) USER STRATEGY CHALLENGES

a: MECHANISM DESIGN CHALLENGES

The crowdsourcing framework includes multiple mechanisms as part of the process. Designing such mechanisms is challenging as it requires considering different aspects as mentioned below.

i) CONFLICTING USER PREFERENCES

Requesters and workers in the framework are of conflicting preferences that need to be considered during task allocation. Requesters are usually interested in the fulfillment of their tasks, with the least possible payment; whereas, workers are interested in the profit acquired from fulfilled tasks. These conflicting preferences may lead to either category of users misbehaving to maximize their gain. Therefore, it is important to account for the different metrics related to the tasks and workers during task allocation.

ii) DISCREPANCY BETWEEN EXPECTED PERFORMANCE AND RECEIVED PERFORMANCE

While task allocation is based on the expected performance of workers for a task, the contribution is important to reflect the received performance of a worker in an allocated task. In some cases, workers with high expected performance do observe a drop in their actual performance. Alternatively, workers with low expected performance might complete tasks with high quality. Therefore, it is important to use mechanisms that evaluate the contribution of workers to pay them their entitled payment and update their reputations and profit accordingly.

iii) WORKER PAYMENT FAIRNESS

With workers of different capabilities performing tasks in the framework, it becomes increasingly important to fairly compute the entitled payment for each of them. In fact, some workers might complete tasks with a quality below what is agreed upon for their payments whether intentionally or due to their limited capabilities. Therefore, the framework needs to determine the entitlement of a worker for their computed payment during and task allocation. Another aspect that hinders the payment fairness is workers colluding to submit a given outcome for the task, thus reducing the payment of honest workers.

iv) MUTUAL REPUTATION VIEW

While requesters can favor workers based on their reputations, workers do not usually have sufficient information to differentiate tasks based on their requester's reputation.

In fact, most of the existing works do not indicate the requester's commitment to their created tasks. Consequently, a requester can cancel the task upon the acceptance of the worker or refuse to pay the worker without prior indication for the worker. This introduces unfairness in the allocation process and the framework. Therefore, it is important to design an allocation mechanism that takes this into account.

b: SECURITY ATTACK CHALLENGES

There are multiple security challenges that hinder a crowdsourcing framework which we describe below.

i) TRUSTED EXECUTION

While a centralized platform is assumed to be trusted, this assumption is not always valid or verifiable. In fact, a major problem in task allocation is the concealed execution of the allocation mechanism since it is hosted on private servers. Hence, the framework can bias the allocation of tasks to increase its profit by colluding with requesters or workers, which hinders the execution trust.

ii) FALSE REPORTING REQUESTERS

Such requesters attempt to report untruthful feedback about a worker's submission to reduce the entitled payment. This is a major challenge to resolve in tasks that cannot be evaluated by the framework itself as, in the long run, it may reduce the interest of workers in engaging with the framework.

iii) PAYMENT BIASING

This can be seen as a result of malicious workers exploiting the framework. Workers can launch a Sybil attack [47] on the framework to bias their payments by impersonating other workers. These malicious workers use the impersonated workers' accounts to submit biased cost values or data to alter the outcome of the payment computation mechanism. Consequently, they can acquire higher payment for their completed tasks. This is an important problem to tackle as it gives workers undeserved payments and consumes the budget from task requesters.

iv) REPUTATION INTEGRITY AND VALIDATION

The reputation of a worker is a dynamically changing metric based on their behavior and is usually updated by the framework. The framework's update for the reputation is assumed to be trusted. However, the concealed execution at the framework makes it possible for the framework to bias the reputation in an untraceable manner. Therefore, there is a need for a methodology to guarantee the integrity of the reputation and to validate its update.

C. THE INTEGRATION OF BLOCKCHAIN AND CROWDSOURCING

The idea of incorporating Blockchain into crowdsourcing frameworks was first introduced in the proposed *PaySense* framework [52], thus opening the way for Blockchain-based crowdsourcing frameworks to emerge [27], [28], [29], [30],

TABLE 1. Summary of surveyed crowdsourcing frameworks.

Paper	Architecture	Task Allocation	Contribution Evaluation	Monetary Payment	Reputation	Application
[12]	Centralized	Optimization (PSO)	Yes	-	Worker	Sensing
[13]			-	Vickrey auction	-	
[14]			-	-	-	
[15]		Genetic algorithm	-	-	Worker	
[16]			-	-	Worker	
[17]		Two-stage auction	-	Yes (bids)	-	
[18]		Quality-driven auction	-	-	-	
[19]		Reverse auction	-	Reverse auction	-	Ride-sharing
[20]		Double auction	Yes (Reliability)	Double auction	-	Sensing
[21]			-	Double auction	-	
[22]			-	Double auction	-	
[23]		Matching Theory	Yes (Ground Truth)	-	Worker	
[24]			-	-	-	
[25]			-	-	-	General
[26]	-		-	-	Ride sharing	
[27]	Blockchain	-	Yes (OFF)	Yes	-	Image labeling
[28]		-	Yes	Yes	-	Sensing
[29]		-	Yes (OFF)	Yes	Worker	Image labeling
[30]		-	Yes (Miners)	Yes	-	Sensing
[31]		Yes (OFF)	-	Yes	-	
[32]		Greedy (ON)	Yes (Similarity)	Yes	Worker/ Requester	
[33]		Auction	Yes (Ground Truth)	Auction	-	
[34]			-	Auction	-	
[35]		Matching Theory	Yes (Requester)	-	Worker	
[36]			-	-	-	
[37]			-	-	-	Ride-sharing
Proposed	Blockchain	Multiple mechanisms/ Context-based	Yes	Yes	Worker/ Requester	General

[31], [32], [33], [34], [35], [36], [37]. The main objective of such frameworks is to overcome some of the challenges in centralized crowdsourcing frameworks more specifically security attack challenges. Alternatively, mechanism design challenges are independent from the use of blockchain. In this section, we describe the contribution of the existing Blockchain-based efforts, to the different aspects of the crowdsourcing process, such as: 1) task allocation, 2) worker's contribution evaluation, 3) payment computation, and 4) feedback and reputation.

c: TASK ALLOCATION

The Blockchain-based crowdsourcing frameworks follow one of two different approaches, i.e. either voluntary selection by workers or through allocation mechanisms. The existing works such as [27], [28], [29], and [30] assume that workers will identify possible tasks and perform them voluntarily in order to maintain the cost-efficiency of the framework and the privacy of workers' information. On the other hand, the works in [31], [32], [35], [36], and [37] utilize task allocation mechanisms to ensure a minimum level of performance quality for allocated tasks. The execution of the allocation mechanism varies where some works perform it using resources off Blockchain (off-chain) as in [31], while other perform it on Blockchain (on-chain). With on-chain execution, the framework guarantees the transparent and autonomous execution of the mechanism, yet the computational complexity of the mechanism becomes important due to the monetary cost implications. Different mechanisms have been explored for on-chain execution such as greedy [32], auction [33], [34],

and stable matching theory [35], [36], [37]. Such mechanisms aim to improve the performance of the framework by improving the allocated pairs while maintaining the cost efficiency of the framework.

d: WORKER CONTRIBUTION EVALUATION

The Blockchain-based frameworks perform the contribution evaluation, either off-chain as in [27], [28], [30], and [31] or on-chain as in [29], [32], [33], and [35]. In [29], a requester-defined evaluation function is used to evaluate the contribution of workers transparently. While transparency is preserved, requesters may bias the submitted evaluation function to reduce the quality of workers' submissions, and consequently their payments or reputations. On the other hand, the work in [32] proposes using an evaluation function embedded as part of the smart contract to compute the contribution of a worker's submission. The evaluation function determines the similarity of the submission to other submitted values.

e: MONETARY PAYMENT

The works in [27], [28], [29], [30], [31], and [32] employ Blockchain to exchange monetary payments in a trusted end-to-end manner between workers and requesters in a crowdsourcing framework. These frameworks share monetary incentives with workers once they complete their assigned tasks, mostly according to the expected worker contribution quality. However, this may lead to overpaying workers who submit low-quality data despite their expected high quality. Other works such as [33] and [34] go beyond

payment exchange and apply auction mechanisms to determine the payment of a worker before exchanging it.

f: Feedback AND REPUTATION

User reputation is considered by a few of the Blockchain-based frameworks to help in determining the eligibility of workers such as [29], [32], and [35]. The works compute the reputation based on a worker's previous interactions and assume the availability of the reputation on Blockchain. The work in [32] goes further and proposes computing the reputation of requesters to provide workers with feedback about task requesters. The work proposes the computation of workers' and requesters' reputations in a smart contract to ensure its integrity.

The existing Blockchain-based crowdsourcing framework elevates the crowdsourcing paradigm by introducing different properties. However, the use of Blockchain entails multiple challenges concerning scalability, security, privacy, integration, regulations, and professional preparations [53]. We further elaborate on two main challenges that need to be considered when designing a crowdsourcing framework on Blockchain being: scalability and computational efficiency, and security.

i) SCALABILITY AND COMPUTATIONAL COMPLEXITY

When deploying a Blockchain-based framework, the scalability of the framework and its ability to properly handle a large number of users and requests is a crucial aspect, especially for near real-time applications. For instance, a centralized crowdsourcing framework such as Uber handles around 18.7 million trips per day,³ thanks to the computational capability of the centralized resource. Meanwhile, the scalability of a Blockchain is limited by the need to maintain the full list of transactions at least at two nodes in the network. The maximum block size and the deployed consensus protocol of the Blockchain affect the processing speed of transactions. In Bitcoin, 7-8 transactions can be processed per second, while for Ethereum it is 20 transactions per second [54]. This limits the applicability of Blockchain-based crowdsourcing frameworks.

Looking at commercial crowdsourcing platforms, these numbers fall below the real demand where Uber managed to handle around 18.7 million trips per day in 2020.⁴ With a limited number of transactions per second, the processing time of transactions varies based on the gas price, computational resources, and consensus mechanism used. Hence, it is important to select the right category of Blockchain being public, private, or consortium to answer the requirements of the intended crowdsourcing framework.

In addition, the monetary cost of executing a mechanism on Blockchain, especially public Blockchain, is directly proportional to its computational complexity. Therefore, the use of optimization techniques for the different components of

the framework becomes challenging and there is a need to select computationally efficient mechanisms that balance the computational complexity and the framework performance. A designed framework needs to account for the distribution of its components between on-chain and off-chain resources as it impacts the computational efficiency of the framework.

ii) SECURITY

The security of the crowdsourcing framework infrastructure directly relates to the security of Blockchain. Unfortunately, Blockchain is vulnerable to multiple security attacks such as the >50% attack [55]. It is possible when the majority of the miners are managed by a single entity or when they maliciously collide to validate transactions for their benefit. In this attack, illegitimate transactions are validated to double-spend coins and reject legitimate transactions to threaten Blockchain's trust. The success of this attack in 2018 led to the loss of more than \$20M worth of cryptocurrency [56]. This attack hinders the trust of a crowdsourcing framework as it would not only lead to the loss of monetary assets but also the injection of incorrect transactions to the record of the framework.

D. DISCUSSION

Table 1 presents a summary of the surveyed crowdsourcing efforts in centralized and Blockchain infrastructures.⁵ It can be seen that the proposed works focus on different stages of crowdsourcing. In addition, each of them focuses on a specific application domain where the adopted mechanisms for allocation, evaluation, and feedback answer the requirements of that application domain. It can be seen that none of the existing works propose a trusted generic framework that overcomes the limitations of centralized frameworks and adapts to different application domains and contexts. Therefore, our main contribution is a crowdsourcing framework that is capable of processing crowdsourced tasks for different applications and varying contexts with high performance, while leveraging Blockchain to provide transparency and verifiable execution to the overall process.

III. PROPOSED BLOCKCHAIN-BASED CROWDSOURCING FRAMEWORK

Fig. 3 shows the novel envisioned Blockchain-based crowdsourcing framework that 1) provides requesters and workers with high performance and trust, and 2) answers different contexts by intelligently switching the used computational mechanisms based on the current context. The main difference between the envisioned framework and existing ones is the consideration of the current context of the environment and the workers, enabling the switch between the allocation different mechanisms, allowing for better task requirements' satisfaction. The context is defined by the supply, demand, and task requirements. The proposed framework is formed from two classes of components: core and advanced.

³<https://www.businessofapps.com/data/uber-statistics/>

⁴<https://www.businessofapps.com/data/uber-statistics/>

⁵OFF= Off-chain, ON= On-chain, - = Not Available.

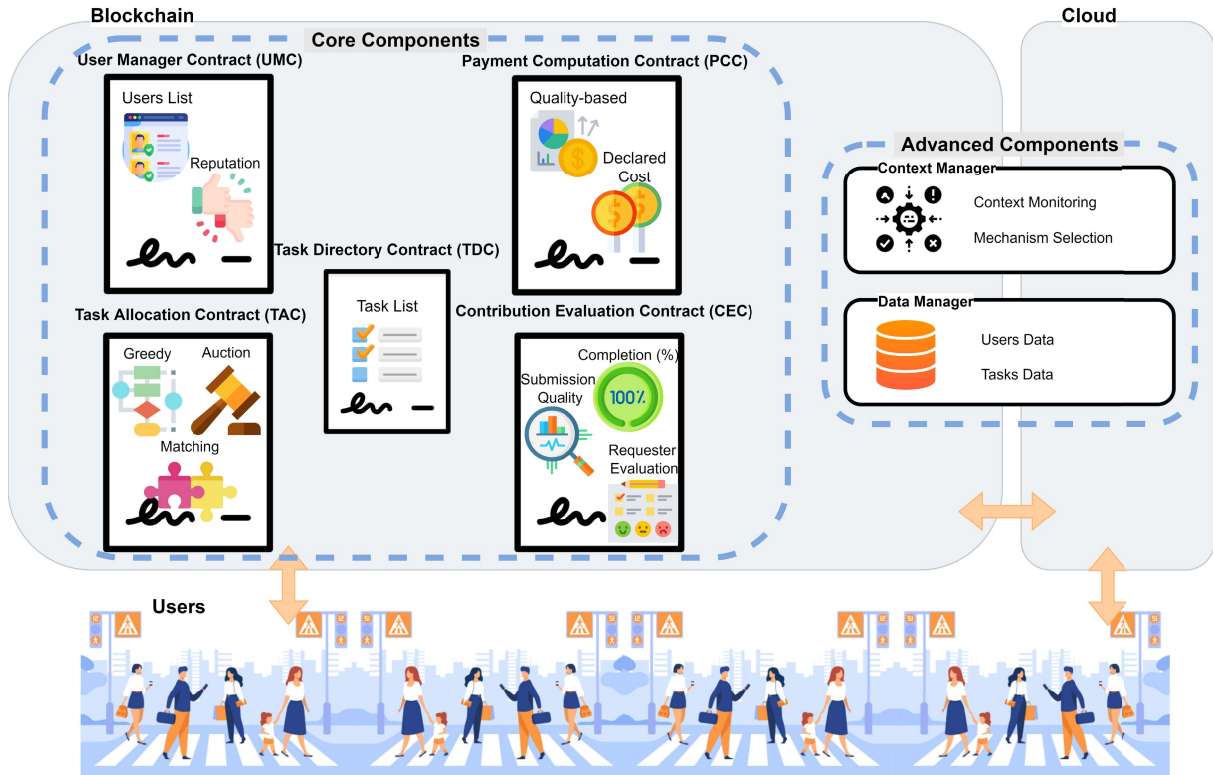


FIGURE 3. Proposed blockchain-based framework showing the users and the framework with the division of its components.

The *core components* are responsible for the basic functionalities of the framework such as task allocation, contribution evaluation, payment computation, and user feedback. They are designed as smart contracts on Blockchain to offer traceable and autonomous execution. The *advanced components* are responsible for context and data management. As these components require computationally expensive mechanisms, they are implemented spanning Blockchain and the cloud to reduce the computations performed on Blockchain by the framework.

A. FRAMEWORK USERS

The users in the framework are the requesters and workers interacting with it. Both categories of users have a reputation value associated with them. For requesters, their reputation is associated with their task cancellation rate as proposed in [32]. This allows workers to differentiate requesters based on their reputations before accepting to perform their tasks. For workers, reputation is associated with their task completion rate proposed in [32].

An additional metric that characterizes workers is the Quality-of-Information (QoI). QoI is a standard metric for crowdsourcing applications used to evaluate the contribution of workers to a task by integrating multiple independent metrics such as distance from the task, completion time, and worker reputation. The QoI considered in this work is the one

proposed in [32].

$$QoI_{wt} = \frac{Rep_w}{D_{wt} \times CT_{wt}} \quad (1)$$

where Rep_w is the reputation of worker w , D_{wt} is the euclidean distance between worker w and task t , and CT_{wt} is the completion time of worker w for task t .

B. BLOCKCHAIN SELECTION

Three categories of Blockchain can be used to construct the Blockchain part of the framework: public, private, or consortium. The different categories vary in their properties where the respective Blockchain can be selected for the framework accordingly.

A public Blockchain is permissionless and open for any user interested to join as an anonymous client or a miner. Therefore, the consensus algorithms used for it require computational effort, time, or stake to verify the blocks due to the lack of trust among the members. Hence, the transaction fee is required to compensate miners' efforts where a higher payment results in a fast confirmation time. A public Blockchain is designed to be immutable where a transaction is irreversible with non-repudiation guaranteed. This category is suitable for crowdsourcing tasks that are not concerned with the privacy of data such as item-selling platforms.

A private Blockchain is permissioned, and users are known members of a single organization. Unlike a public Blockchain, transactions are verified by selected miners from

TABLE 2. Blockchain categories based on privacy, org. implies an organization.

Metric	Public	Private	Consortium
Permissions	Permissionless	Permissioned	
Users	Any user	One org.	Multiple org.
Miners	Anyone	Approved miners	
Transaction fees	Yes	No	No
Confirmation speed	Low	High	High
Immutability	Yes		Partial

within the same organization. Consensus protocols such as Practical Byzantine Fault Tolerance (PBFT) and Tendermint are designed for it with the transaction speed being considerably high while the transaction fee is negligible. Unfortunately, a private Blockchain is not immutable as it is managed by a centralized organization. Thus, the organization can roll back its chain to an earlier block removing transactions from the chain. This category is appropriate for internal use by the framework and tasks concerned with data privacy such as collected data.

A consortium Blockchain is an intermediary between the previously mentioned types of Blockchain. It is a permissioned Blockchain with the properties of private Blockchain in terms of miners and consensus protocols. However, the difference is that members who are granted permission to access it or allocated as miners can be from different organizations. Immutability, in this case, is partially preserved as a single organization in the consortium cannot roll back the chain. However, if the majority of the organizations agree, the current chain can be tampered with or rolled back. A consortium Blockchain fits crowdsourced supply chain frameworks as multiple entities are part of the crowdsourcing process. Table 2 presents a summary of the characteristics of each Blockchain type. In the proposed framework, the public Blockchain is considered.

C. CORE COMPONENTS- SMART CONTRACTS

The core components with their respective computational mechanisms implemented as smart contracts and hosted on Blockchain are shown in Fig. 3. This ensures that the mechanisms are highly available, transparently executed, cost-efficient, and open to users to utilize the framework capabilities. Different computational mechanisms are implemented for each stage to equip the framework with sufficient ones for different possible contexts.

Table 3 shows the **User Manager Contract (UMC)** responsible for maintaining the information of workers and requesters and compiling their feedback and reputations.

The *User* data structure is designed to hold a user's information with its different fields. UMC stores users' information in the *User List* mapping, which maps a user's address to their *User* object. *City Workers* maps the city code to the addresses of available workers within the city. City codes are used for the mapping, as opposed to latitude and longitude coordinates since they change less frequently, making them a more cost-efficient choice when Blockchain is used. The *addUser()* function allows a user to register by providing the

TABLE 3. User manager contract (UMC).

Data Structure		
User		
Completion Time (uint)	Reputation (uint)	Role (bytes1)
Total Tasks Counter (uint)	Latitude (uint)	Longitude (uint)
Worker Availability (boolean)	Accepted/ Cancelled Tasks Counter (uint)	Cities (bytes1[])
Variables		
User List (address \Rightarrow User)		
City Workers (bytes1 \Rightarrow address[])		
Function	Parameters	Return
<i>addUser()</i>	User Information	-
<i>updateWorkerStatus()</i>	Status	-
<i>updateCity()</i>	City, Action	-
<i>updateLocation()</i>	Location	-
<i>updateReputation()</i>	Task Status	-
<i>getWorkers()</i>	City	User[]

TABLE 4. Task directory contract (TDC).

Data Structure		
Task		
Requester (Address)	Reputation (uint)	Duration (uint)
Latitude (uint)	Longitude (uint)	Status (uint)
Deposit (uint)	Min. Reputation (uint)	
Variables		
City Tasks (bytes1 \Rightarrow Task[])		
Active Cities (bytes1[])		
Function	Parameters	Return
<i>addTask()</i>	Task attributes and budget	-
<i>updateTaskStatus()</i>	Status	-

necessary information for a *User* object to be created and mapped in *User List* and *City Workers*. The *updateWorkerStatus()*, *updateCities()*, and *updateLocation()* functions allow workers to update their information. The *updateReputation()* is an internal function called to update the reputation of a requester/ worker according to the adopted mechanism such as the one proposed in [32]. Meanwhile, the *getWorkers()* function is used to acquire the list of workers in a specific city.

Table 4 presents the **Task Directory Contract (TDC)**, which is responsible for storing available tasks.

The *Task* data structure holds the information of a single task. TDC maintains the information of tasks in the *City Tasks* mapping, which maps a city code to an array of tasks within the city. *Active Cities* holds the list of cities with available tasks. TDC includes the *addTask()* and *updateTaskStatus()* functions. The *addTask()* function allows requesters to publish their task to the framework by specifying the task attributes and transferring the intended budget. The function creates a *Task* object, initializing the default values and appending them to the corresponding city in the mapping. The *updateTaskStatus()* function is used for updating the current status of a task, whether pending or completed.

Table 5 presents the **Task Allocation Contract (TAC)**, which is responsible for allocating tasks to workers according to the set mechanism.

TABLE 5. Task allocation contract (TAC).

Variables		
Task-Worker Allocation (address⇒address[])		
Greedy (boolean)	Auction (boolean)	Matching (boolean)
Function	Parameters	Return
<i>GreedyMechanism()</i>	City Code/ Task address	-
<i>AuctionMechanism()</i>	City Code/ Task address	-
<i>MatchingMechanism()</i>	City Code/ Task address	-
<i>UpdateUsedMechanism()</i>	Selected Mechanism	-
<i>AllocateTasks()</i>	City Code/ Task address	-

TABLE 6. Contribution evaluation contract (CEC).

Variables		
Worker Submission (address⇒int)		
Worker Evaluation (address⇒int)		
Completion (boolean)	Quality (boolean)	Requester (boolean)
Function	Parameters	Return
<i>EvaluateCompletion()</i>	Task address	-
<i>EvaluateQuality()</i>	Task address	-
<i>AddRequesterEvaluation()</i>	Worker address, evaluation	-
<i>UpdateUsedMechanism()</i>	Selected Mechanism	-
<i>ContributionEvaluation()</i>	Task address	-

The *Task-Worker Allocation* mapping maps the address of a task to an array of allocated workers for it. In addition, TAC includes multiple boolean variables that indicate the currently used mechanism among the ones hosted within the smart contract. Each of the algorithms is presented by its respective function being *GreedyMechanism()*, *AuctionMechanism()*, and *MatchingMechanism()*. The functions use the list of workers and tasks for the allocation based on the currently used mechanism. Consequently, it sets the *task-worker allocation* mapping. The *UpdateUsedMechanism()* function is triggered to set the used function by setting its corresponding variable. The *AllocateTasks()* function performs the allocation by checking the variables and running the corresponding used mechanism.

Table 6 presents the **Contribution Evaluation Contract (CEC)**, which is responsible for evaluating the contribution of each worker by the end of an allocated task.

The contract holds two mappings: *Worker Submission* and *Worker Evaluation*. The *Worker Submission* maps a worker’s address to their submitted value for the task, while *Worker Evaluation* maps the worker’s address to their calculated evaluation result. CEC holds multiple boolean variables that reflect which evaluation mechanism is used. The *EvaluateCompletion()* function evaluates whether a worker has completed their allocated tasks or not. The *EvaluateQuality()* function computes the quality of a worker’s submission for tasks based on similarity, majority voting, etc. The *AddRequesterEvaluation()* function allows a requester to submit an evaluation for a given task. The *UpdateUsedMechanism()* function is used to set the used evaluation mechanism.

TABLE 7. Payment computation contract (PCC).

Variables		
Worker Cost (address⇒int)		
Requester Budget (address⇒int)		
Completion (boolean)	Quality (boolean)	Requester (boolean)
Function	Parameters	Return
<i>AddWorkerCost()</i>	Cost	-
<i>AddRequesterCost()</i>	Cost	-
<i>QualityPayment()</i>	Evaluation	-
<i>AuctionPayment()</i>	Evaluation	-
<i>UpdateUsedMechanism()</i>	Selected Mechanism	-
<i>PaymentComputation()</i>	Task address	-

ContributionEvaluation() function executes the set contribution evaluation function.

Table 7 presents the **Payment Computation Contract (PCC)**, which calculates and distributes the payments on workers.

The contract stores workers’ submitted costs and requesters’ submitted budgets in the respective mappings, *Worker Cost* and *Requester Budget*. The *AddWorkerCost()* function allows a worker to submit their required payment to perform an allocated task. On the other hand, the *AddRequesterCost()* function allows a requester to declare the maximum possible payment for a task. The *QualityPayment()* function computes the payments for workers based on their contribution and the budget of the task, a mechanism similar to the one presented in [32]. The *AuctionPayment()* function calculates the payment based on an adopted auction mechanism accounting for the declared costs by the worker and requesters similar to the work in [34]. The *PaymentComputation()* function calculates workers’ payments according to the selected computation mechanism and forwards the payments to the entitled workers from the deposited budget. The *UpdateUsedMechanism()* function sets the used payment mechanism by setting the corresponding boolean variable.





D. ADVANCED COMPONENTS

The proposed framework has advanced components that are part of its architecture. Two managers are proposed: 1) **Context Manager** and 2) **Data Manager**. While the basic components are deployed fully on-chain, the advanced components span resources on Blockchain and the cloud to benefit from both of these architectures’ capabilities. Blockchain provides the managers with transparency and trust. However, it is computationally expensive to perform all the context inference and store the data on the Blockchain, as it all translates to monetary cost. Therefore, cloud resources are employed to migrate mechanisms for cost efficiency.

The **Context Manager** is responsible for monitoring and evaluating the current context to infer the suitable mechanisms for the core components.

In this work, the context is defined by the current supply (workers) and demand (tasks) as well as the requirements of the tasks shown in Table 8. The concept of context manager is already implemented in centralized crowdsourcing. Its role

TABLE 8. Possible contexts and the respective mechanisms at the process stages.

Application Domain	Environment Monitoring	Delivery	Ride Sharing	Emergency and Health Crisis
				
Context Information				
Worker-Task Requirements	Many-One	One-One, One-Many	One-One	One-One
Time-critical	Yes/No	Yes/No	Yes	Yes
Location Dependent	Yes	Yes	Yes	Yes
Demand-to-Supply Ratio (DSR)	Yes	Yes	Yes	NA
Fulfillment vs Quality	Both	Both	Fulfillment	Fulfillment
Suggested Mechanisms				
Task Allocation	Greedy (Delay tolerant), GSM (High DSR)	Auction (Declared cost)	GSM (High DSR), Greedy (Low DSR)	Greedy (For timeliness)
Contribution Evaluation	Submission Quality	Task Completion	Requester Evaluation	Task Completion
Payment Computation	Quality-based	Declared Cost	Declared Cost	Fixed Budget
Worker/ Requester Feedback	Evaluation	Task Completion	Requester Evaluation	Task Completion

is to monitor the current context of the framework being the number of tasks and workers as well as other metrics and adjust the used algorithms in the framework components for task allocation, payment, etc to maximize its performance and users’ satisfaction. Uber⁶ is an example of a crowdsourcing framework that applies the surge factor to balance the load-based on the supply and demand on the platform.

This manager serves two main functions: 1) context monitoring and 2) mechanism selection. The context monitoring component aims to record relevant metrics for the current context being the number of tasks (demand), the number of workers (supply), and task requirements. These metrics can be collected periodically from Blockchain since it holds users’ interactions and the published information about the current tasks as transparent transactions. These collected metrics can be further used in mechanism selection.

The mechanism selection component aims to determine the mechanisms to employ based on the current context. The selection can be done through statistical methods. Emerging technologies such as Artificial Intelligence (AI) and Machine Learning (ML) can be also employed. ML allows utilizing the collected context information to train ML models that predict the most suitable mechanisms to deploy given the current context. The computational cost of determining the context of a task depends on the ML model used and where is it applied (cloud or blockchain). If the ML model is to be hosted on blockchain, then its computational cost needs to be minimized. However, if the ML model is to be hosted on the Cloud, then more complex models can be used. Upon the prediction of the models, the smart contracts’ *updateUsedMechanisms()* functions can be invoked to set the corresponding mechanisms. The ML models can be trained and hosted on the cloud or Blockchain, yet the cloud poses a cost-efficient option.

Table 8 presents the context for a few application domains that the framework can consider. In addition, it presents suggested mechanisms for each category of tasks. It can be

seen that tasks in distinct applications vary in their time and location dependencies. In addition, they differ in their requirements for the number of workers and their sensitivity to change in Demand-to-Supply Ratio (DSR) as well as quality requirements. Some tasks require being fulfilled without constraints on the quality while other tasks required fulfillment with high-quality constraints. In addition to the above domains, the proposed framework can be used to answer tasks for AI and ML models applicable for different application domains. Workers can be selected by the framework to collect data required by the ML models.

The **Data Manager** deals with the generated data by the crowdsourcing system such as users’ profiles, and the crowd collected data. It is responsible for processing and aggregating the collected data before being pushed back to the requesters of the tasks. In requires determining cost-efficient storage and a mechanism to out-date data that cannot be reused. Crowdsourced data can be stored either on the Blockchain or the cloud. It is worth noting that storage on the Blockchain is of higher cost and slower access time than storage on the cloud. Therefore, the cloud is used to store huge amounts of data in a private and cost-efficient manner. Consequently, a small quantity of data can be stored on the Blockchain, which in turn overcomes the storage scalability constraint of the public Blockchain.

IV. A CASE STUDY ON TASK ALLOCATION MECHANISMS FOR BLOCKCHAIN-BASED FRAMEWORKS

The proposed framework is a holistic Blockchain-based crowdsourcing framework that accommodates tasks from various contexts. It is envisioned that the framework will intelligently switch between the different mechanisms based on the monitored context. In this evaluation, three state-of-the-art task allocation algorithms for Blockchain-based crowdsourcing frameworks are compared to understand their performance under different contexts. The selected mechanisms perform on-chain task allocation and are hosted on Blockchain, which aligns with the objective of the proposed framework. Other surveyed state-of-the-art Blockchain

⁶<https://www.uber.com/us/en/drive/driver-app/how-surge-works/>

works were excluded from the evaluation as they either do not perform allocation on Blockchain such as [33] or their allocation mechanisms are tailored to answer the requirements of a more specific objective such as privacy such as the works in [31], [35], and [36]. The considered mechanisms are-

- 1) SenseChain [32] which performs a greedy allocation of tasks to workers based on a proposed QoI metric, maximizing its value for allocated pairs. Workers submit their task of interest, one task at a time for each allocation round.
- 2) Repeated task allocation mechanism proposed in [34] and [57] which utilize an auction mechanism to allocate tasks based on workers' bids. A worker can submit a bid for one task or more every repetition of the allocation. Repeated-Single-Minded Bidder (R-SMB) is referred to in this section.
- 3) Gale-Shapley Matching (GSM) [37], which employs a matching mechanism to account for the preferences of the workers and requesters during task allocation. Workers determine their task preferences based on a proposed Quality-of-Task (QoT) metric, while requesters determine their worker preferences based on a QoI metric.

The details for the allocation mechanisms can be found in their respective references.

A. SIMULATION SETUP AND PARAMETERS

The allocation mechanisms were implemented on Matlab 2020b to compare their performance. A real dataset collected from the Xively platform⁷ with around 500 workers at random locations was used. The dataset includes the IDs of the workers and their locations. Out of them, a set of randomly selected workers was identified. These workers' reputations were generated as uniformly distributed random values and appended to the dataset. In addition, a random set of tasks was generated within the area of the workers. The sets of workers and tasks were used as an input to the allocation mechanisms. The obtained results were compared to evaluate the performance of each in several contexts.

Table 9 outlines the evaluation setup as well as the parameters for the dataset used in the evaluation and the generated tasks. In addition, it presents the parameters for the task allocation mechanisms considered in the evaluation.

B. PERFORMANCE EVALUATION AND COMPARISON

The evaluation aims to understand the performance of the different mechanisms in terms of the percentage of allocated tasks, their allocation time, allocated workers' QoI, and the traveled distance by allocated workers. The change in context in the evaluation is implied by the different DSRs presented in this section.

Fig. 4 illustrates the percentage of allocated tasks which reflects the fulfillment of the tasks. Each allocation algorithm is repeated multiple times to account for unallocated tasks and

TABLE 9. Simulation parameters.

Experimental Setup	
Tool	Matlab 2020b
DSR	[0.1,0.1,10]
Number of Iteration	5
Worker Dataset Attributes	
Number of workers	100
Location (Latitude, Longitude)	([35, ..., 44],[136, ..., 142])
Data Type	Radiation Level (μ SV)
Workers reputation (Rep)	[40, ..., 70]
Worker's radius of interest (r)	2
Task Generation Parameters	
Number of Tasks (N)	[10, 20, ..., 1000]
Location (Latitude, Longitude)	([35, ..., 44],[136, ..., 142])
Number of workers per tasks (n)	1
Greedy Mechanism Parameters	
Preference list length	1
Worker Selection Criteria	QoI in [32]
Task Selection Criteria	QoI in [32]
Auction Mechanism Parameters	
Preference list length	1
Worker Selection Criteria	Distance
Task Selection Criteria	Worker Rank
Matching Mechanism Parameters	
Preference list length	up to 10
Worker Selection Criteria	QoT in [37]
Task Selection Criteria	QoI in [32]

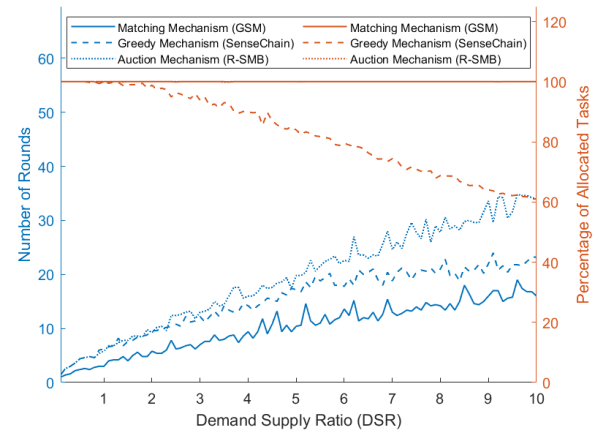


FIGURE 4. Percentage of allocated tasks (red) and number of rounds (blue).

the algorithm is terminated when either all tasks are allocated or no task is feasible for available workers. The number of repetitions reflects the time required for the allocation and is referred to as the number of rounds. The number of rounds required by each mechanism is also presented in Fig. 4.

GSM and R-SMB maximize the percentage of allocated tasks independent from the DSR value where the results overlap in the figure shown. Meanwhile, SenseChain performs similar to the other mechanisms at DSR values below 2 in the task allocation percentage while the allocation percentage is much lower at higher DSR values.

GSM converges to the maximum task allocation percentage within a lower number of rounds than R-SMB. The gap between GSM and R-SMB, in the number of rounds,

⁷<https://eprints.soton.ac.uk/354861/3/XivelyData.csv>

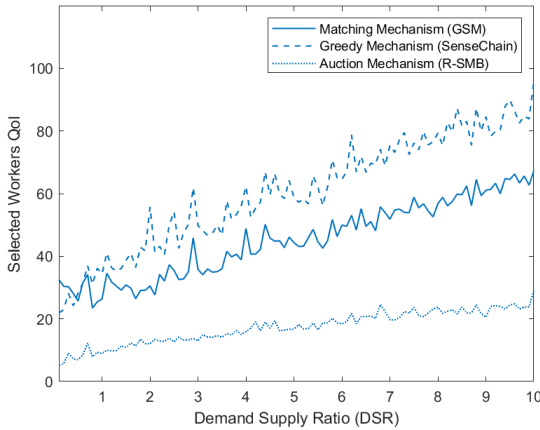


FIGURE 5. Average workers QoI.

increases with the reduction in the competition between workers for available tasks (high DSR values). GSM requires fewer rounds as workers submit multiple preferences at each round, compared to the other approaches where one preference is submitted. For both mechanisms, the number of rounds increases with the increase in DSR as the workers' preferences are more spread due to the drop in the competition between workers. On the other hand, SenseChain terminates the allocation at a smaller number of rounds than R-SMB at DSR values greater than 2. However, the termination is linked to a lower task allocation percentage, which would affect the performance of the framework.

Fig. 5 shows the average QoI of allocated workers by the end of the allocation rounds. At high competition context (DSR less than 1), GSM outperforms the other two mechanisms with a big difference compared to R-SMB as QoI is not part of the selection criteria in the latter mechanism. The performance of SenseChain consistently improves with the drop in the competition between workers while the other mechanisms do not increase with the same proportion. SenseChain provides the best QoI compared to the alternative mechanisms as QoI is used at the worker and the task for the allocation. GSM follows as QoI is only considered in the task allocation while the worker selection uses the QoT metric.

Fig. 6 shows the average travelled distance by allocated workers, which is proportional to the time needed to perform the task. The results show that SenseChain performs well in the high competition context (DSR less than 0.2) and minimizes the traveled distance, hence the time to perform an allocated task. However, the average traveled distance increases logarithmically with the reduction of the competition between workers due to the admission of workers at further locations to perform tasks. The performance of GSM and R-SMB improves as the distance reduces with the reduction in the competition, and they converge to a similar value for the traveled distance. GSM and R-SMB are of small distance as they both incorporate it in the allocation. It was expected that R-SMB results in the smallest traveled distance, but the small additional difference compared to GSM is due

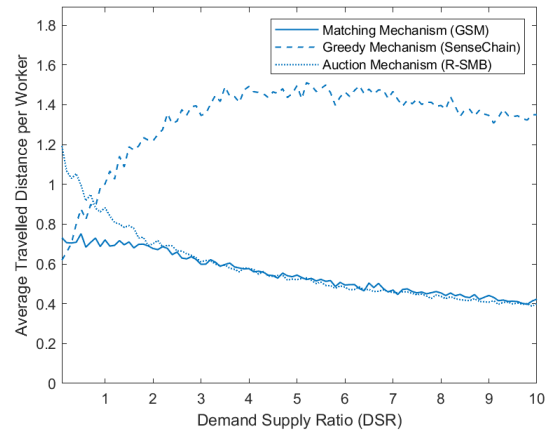


FIGURE 6. Average travelled distance.

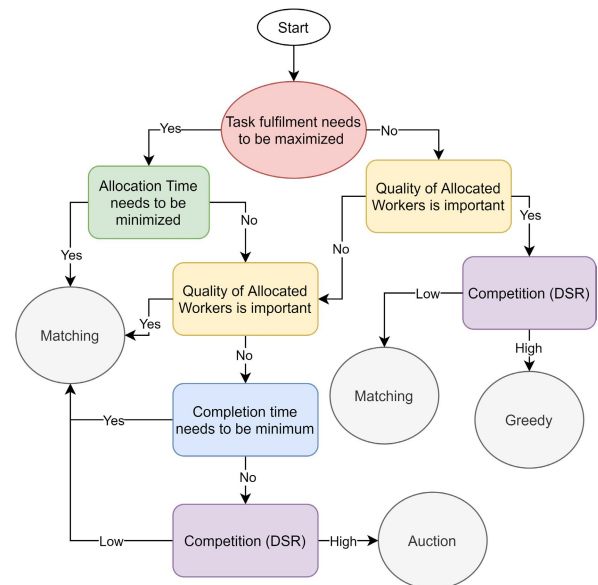


FIGURE 7. Allocation mechanism flow.

to the additional rounds that admit further workers to the allocation. SenseChain focuses on QoI, which incorporates other metrics, leading to longer distances. It is inferred from the figure that GSM can converge to a smaller distance value at the different DSR values.

C. EVALUATION SUMMARY

Fig. 7 presents a logical map that reflects the usability context for the different mechanisms based on the observed results. The greedy mechanism is best when the quality of allocated workers is important and the competition in the framework is low as in the case of environmental monitoring applications. However, such allocation does not maximize either the fulfillment or the allocation time. On the other hand, the auction mechanism is best when the fulfillment needs to be maximized while sacrificing the allocation time and the completion time under high competition contexts as in the case of delivery applications. In alternative contexts, the results demonstrated that the stable matching mechanism

performs best for the measured metrics such as ride-sharing and environment monitoring under high DSR.

V. CONCLUSION

In this paper, the current crowdsourcing frameworks are discussed and the challenges and opportunities in general and blockchain-based crowdsourcing framework are identified. Additionally, a novel context-aware Blockchain-based crowdsourcing framework is proposed. It consists of core components hosted on Blockchain and advanced components hosted spanning Blockchain and the cloud for flexibility and scalability. The framework holds a manager responsible for updating the used mechanisms according to the context. Finally, a use case study for the possible task allocation algorithms is presented using a real dataset. The evaluation aimed to assess the performance of the algorithms under different contexts. The results demonstrate the applicability of each algorithm in different demand to supply ratios. In future work, different machine learning models for context switching can be studied to understand their impact on the performance of the proposed framework.

REFERENCES

- [1] J. Howe, "The rise of crowdsourcing," *Wired Mag.*, vol. 14, no. 6, pp. 1–4, Jun. 2006.
- [2] A. Ghermandi and M. Sinclair, "Passive crowdsourcing of social media in environmental research: A systematic map," *Global Environ. Change*, vol. 55, pp. 36–47, Mar. 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0959378018309920>
- [3] J. Picaut, N. Fortin, E. Bocher, G. Petit, P. Aumond, and G. Guillaume, "An open-science crowdsourcing approach for producing community noise maps using smartphones," *Building Environ.*, vol. 148, pp. 20–33, Jan. 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0360132318306747>
- [4] A. Leiter, T. Sablinski, M. Diefenbach, M. Foster, A. Greenberg, J. Holland, W. K. Oh, and M. D. Galsky, "Use of crowdsourcing for cancer clinical trial development," *J. Nat. Cancer Inst.*, vol. 106, no. 10, Oct. 2014, Art. no. dju258, doi: [10.1093/jnci/dju258](https://doi.org/10.1093/jnci/dju258).
- [5] E. Losina, G. L. Michl, K. C. Smith, and J. N. Katz, "Randomized controlled trial of an educational intervention using an online risk calculator for knee osteoarthritis: Effect on risk perception," *Arthritis Care Res.*, vol. 69, no. 8, pp. 1164–1170, Aug. 2017, doi: [10.1002/acr.23136](https://doi.org/10.1002/acr.23136).
- [6] M. Chaudhary, A. Bansal, D. Bansal, B. Raman, K. K. Ramakrishnan, and N. Aggarwal, "Finding occupancy in buses using crowdsourced data from smartphones," in *Proc. 17th Int. Conf. Distrib. Comput. Netw.*, New York, NY, USA, Jan. 2016, p. 35, doi: [10.1145/2833312.2833460](https://doi.org/10.1145/2833312.2833460).
- [7] P. Mukheja, M. Kiran K, N. R. Velaga, and R. B. Sharmila, "Smartphone-based crowdsourcing for position estimation of public transport vehicles," *IET Intell. Transp. Syst.*, vol. 11, no. 9, pp. 588–595, Nov. 2017.
- [8] W. Wang, Z. He, P. Shi, W. Wu, Y. Jiang, B. An, Z. Hao, and B. Chen, "Strategic social team crowdsourcing: Forming a team of truthful workers for crowdsourcing in social networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 6, pp. 1419–1432, Jun. 2019.
- [9] J. Chamberlain, "Groupsourcing: Distributed problem solving using social networks," in *Proc. 2nd AAAI Conf. Hum. Comput. Crowdsourcing*, Sep. 2014, pp. 1–8.
- [10] L. Hetmank, "Components and functions of crowdsourcing systems—A systematic literature review," *Tech. Rep.*, 2013, pp. 55–69.
- [11] J. Fullerton. (2019). *Uber's China Problem*. [Online]. Available: https://www.vive.com/en_us/article/3daa55/ubers-china-problem
- [12] R. Estrada, R. Mizouni, H. Otrok, A. Ouali, and J. Bentahar, "A crowdsensing framework for allocation of time-constrained and location-based tasks," *IEEE Trans. Services Comput.*, vol. 13, no. 5, pp. 769–785, Sep. 2020.
- [13] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 6, pp. 1330–1341, Jun. 2019.
- [14] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "ActiveCrowd: A framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Trans. Hum.-Mach. Syst.*, vol. 47, no. 3, pp. 392–403, Jun. 2017.
- [15] R. Azzam, R. Mizouni, H. Otrok, A. Ouali, and S. Singh, "GRS: A group-based recruitment system for mobile crowd sensing," *J. Netw. Comput. Appl.*, vol. 72, pp. 38–50, Sep. 2016, doi: [10.1016/j.jnca.2016.06.015](https://doi.org/10.1016/j.jnca.2016.06.015).
- [16] M. Abououf, R. Mizouni, S. Singh, H. Otrok, and A. Ouali, "Multi-worker multi-task selection framework in mobile crowd sourcing," *J. Netw. Comput. Appl.*, vol. 130, pp. 52–62, Mar. 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1084804519300086>
- [17] Y. Wang, Z. Cai, Z. Zhan, Y. Gong, and X. Tong, "An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 3, pp. 414–429, Apr. 2019.
- [18] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, Y. Cheng, and X. Shen, "Quality-driven auction-based incentive mechanism for mobile crowd sensing," *IEEE Trans. Veh. Technol.*, vol. 64, no. 9, pp. 4203–4214, Sep. 2015.
- [19] M. Xiao, K. Ma, A. Liu, H. Zhao, Z. Li, K. Zheng, and X. Zhou, "SRA: Secure reverse auction for task assignment in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 4, pp. 782–796, Apr. 2020.
- [20] H. Jin, L. Su, and K. Nahrstedt, "CENTURION: Incentivizing multi-requester mobile crowd sensing," 2017, *arXiv:1701.01533*.
- [21] Y. Wei, Y. Zhu, H. Zhu, Q. Zhang, and G. Xue, "Truthful online double auctions for dynamic mobile crowdsourcing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2074–2082.
- [22] H. Huang, Y. Xin, Y.-E. Sun, and W. Yang, "A truthful double auction mechanism for crowdsensing systems with max-min fairness," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.
- [23] S. Yang, K. Han, Z. Zheng, S. Tang, and F. Wu, "Towards personalized task matching in mobile crowdsensing via fine-grained user profiling," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 2411–2419.
- [24] J. Shu, X. Liu, Y. Zhang, X. Jia, and R. H. Deng, "Dual-side privacy-preserving task matching for spatial crowdsourcing," *J. Netw. Comput. Appl.*, vol. 123, pp. 101–111, Dec. 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1084804518302959>
- [25] J. Shu, X. Liu, X. Jia, K. Yang, and R. H. Deng, "Anonymous privacy-preserving task matching in crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3068–3078, Aug. 2018.
- [26] M. Abououf, S. Singh, H. Otrok, R. Mizouni, and A. Ouali, "Gale-Shapley matching game selection—A framework for user satisfaction," *IEEE Access*, vol. 7, pp. 3694–3703, 2018.
- [27] Y. Lu, Q. Tang, and G. Wang, "ZebraLancer: Decentralized crowdsourcing of human knowledge atop open blockchain," 2018, *arXiv:1803.01256*.
- [28] M. Arafeh, M. E. Barachi, A. Mourad, and F. Belqasmi, "A blockchain based architecture for the detection of fake sensing in mobile crowdsensing," in *Proc. 4th Int. Conf. Smart Sustain. Technol. (SpliTech)*, Jun. 2019, pp. 1–6.
- [29] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1251–1266, Jun. 2019.
- [30] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, p. 17545–17556, 2018.
- [31] M. Yang, T. Zhu, K. Liang, W. Zhou, and R. H. Deng, "A blockchain-based location privacy-preserving crowdsensing system," *Future Gener. Comput. Syst.*, vol. 94, pp. 408–418, May 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X18320909>
- [32] M. Kadadha, H. Otrok, R. Mizouni, S. Singh, and A. Ouali, "Sensechain: A blockchain-based crowdsensing framework for multiple requesters and multiple workers," *Future Gener. Comput. Syst.*, vol. 105, pp. 650–664, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X19312579>
- [33] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui, "Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain," in *Proc. 15th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2018, pp. 442–450.
- [34] M. Kadadha, R. Mizouni, S. Singh, H. Otrok, and A. Ouali, "ABCrowd an auction mechanism on blockchain for spatial crowdsourcing," *IEEE Access*, vol. 8, pp. 12745–12757, 2020.

- [35] J. An, H. Yang, X. Gui, W. Zhang, R. Gui, and J. Kang, "TCNS: Node selection with privacy protection in crowdsensing based on twice consensus of blockchain," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 1255–1267, Sep. 2019.
- [36] Y. Wu, S. Tang, B. Zhao, and Z. Peng, "BPTM: Blockchain-based privacy-preserving task matching in crowdsourcing," *IEEE Access*, vol. 7, pp. 45605–45617, 2019.
- [37] M. Kadadha, H. Otrok, S. Singh, R. Mizouni, and A. Ouali, "Two-sided preferences task matching mechanisms for blockchain-based crowdsourcing," *J. Netw. Comput. Appl.*, vol. 191, Oct. 2021, Art. no. 103155. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1084804521001697>
- [38] M. Kadadha and H. Otrok, "A blockchain-enabled relay selection for QoS-OLSR in urban VANET: A Stackelberg game model," *Ad Hoc Netw.*, vol. 117, Jun. 2021, Art. no. 102502. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570870521000615>
- [39] D. Dujak and D. Sajter, *Blockchain Applications in Supply Chain*. Cham, Switzerland: Springer, 2019, pp. 21–46, doi: [10.1007/978-3-319-91668-2_2](https://doi.org/10.1007/978-3-319-91668-2_2).
- [40] A. D. Dwivedi, R. Singh, S. Dhall, G. Srivastava, and S. K. Pal, "Tracing the source of fake news using a scalable blockchain distributed network," in *Proc. IEEE 17th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Dec. 2020, pp. 38–43.
- [41] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, Oct. 2009. [Online]. Available: <https://metzdowd.com>
- [42] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, 2014.
- [43] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2016, pp. 254–269, doi: [10.1145/2976749.2978309](https://doi.org/10.1145/2976749.2978309).
- [44] J. Redman. (2020). *Developer Launches MTURK Alternative 'Taskopus' Powered by Bitcoin Cash | Sharing Economy Bitcoin News*. [Online]. Available: <https://news.bitcoin.com/developer-launches-mturk-alternative-taskopus-powered-by-bitcoin-cash/>
- [45] D. Govindaraj, K. V. M. Naidu, A. Nandi, G. Narlikar, and V. Poosala, "MoneyBee: Towards enabling a ubiquitous, efficient, and easy-to-use mobile crowdsourcing service in the emerging market," *Bell Labs Tech. J.*, vol. 15, no. 4, pp. 79–92, Mar. 2011.
- [46] K. Wazny, "Applications of crowdsourcing in health: An overview," *J. Global Health*, vol. 8, no. 1, Jun. 2018, Art. no. 010502. [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/29564087>
- [47] J. R. Douceur, "The Sybil attack," in *Proc. Int. Workshop Peer-Peer Syst.* Berlin, Germany: Springer-Verlag, 2002, pp. 251–260.
- [48] L. de Alfaro, A. Kulshreshtha, I. Pye, and B. T. Adler, "Reputation systems for open collaboration," *Commun. ACM*, vol. 54, no. 8, pp. 81–87, Aug. 2011, doi: [10.1145/1978542.1978560](https://doi.org/10.1145/1978542.1978560).
- [49] C. Burnett, T. J. Norman, and K. Sycara, "Bootstrapping trust evaluations through stereotypes," in *Proc. 9th Int. Conf. Auto. Agents Multiagent Syst.*, Richland, SC, USA, vol. 1, 2010, pp. 241–248.
- [50] D. Alishev, R. Hussain, W. Nawaz, and J. Lee, "Social-aware bootstrapping and trust establishing mechanism for vehicular social networks," in *Proc. IEEE 85th Veh. Technol. Conf. (VTC Spring)*, Jun. 2017, pp. 1–5.
- [51] O. A. Wahab, R. Cohen, J. Bentahar, H. Otrok, A. Mourad, and G. Rjoub, "An endorsement-based trust bootstrapping approach for newcomer cloud services," *Inf. Sci.*, vol. 527, pp. 159–175, Jul. 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025520302838>
- [52] S. Delgado-Segura, C. Tanas, and J. Herrera-Joancomartí, "Reputation and reward: Two sides of the same bitcoin," *Sensors*, vol. 16, no. 6, p. 776, May 2016.
- [53] J. Al-Jaroodi and N. Mohamed, "Blockchain in industries: A survey," *IEEE Access*, vol. 7, pp. 36500–36515, 2019.
- [54] H. Tang, Y. Shi, and P. Dong, "Public blockchain evaluation using entropy and TOPSIS," *Expert Syst. Appl.*, vol. 117, pp. 204–210, Mar. 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0957417418306250>
- [55] A. Baliga. (2019). *Understanding Blockchain Consensus Models*. Persistent. [Online]. Available: <https://pdfs.semanticscholar.org/da8a/37b10bc1521a4d3de925d7ebc44bb606d740.pdf>
- [56] B. Dale. (2019). *This College Freshman is Out to 51% Attack Your Cryptocurrency—CoinDesk*. [Online]. Available: <https://www.coindesk.com/this-college-freshman-is-out-to-51-attack-your-cryptocurrency>

- [57] L. Gao, T. Cheng, and L. Gao, "TSWCrowd: A decentralized task-select-worker framework on blockchain for spatial crowdsourcing," *IEEE Access*, vol. 8, pp. 220682–220691, 2020.



MAHA KADADHA (Member, IEEE) received the B.Sc. degree in electrical and computer engineering and the M.Sc. degree in computer engineering from the Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, and the Ph.D. degree in electrical and computer engineering from the Khalifa University of Science and Technology, Abu Dhabi. She is currently a Postdoctoral Fellow with the Department of ECE, Khalifa University of Science and Technology. She is also an Active Reviewer at: *IEEE Communications Magazine*, *IEEE INTERNET OF THINGS JOURNAL*, and *IEEE Network* magazine. Her research interests include crowd sensing/sourcing, blockchain, and wireless sensor networks.



SHAKTI SINGH received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, Khalifa University of Science and Technology, Abu Dhabi, UAE. His research interests include semiconductor devices and integrated circuits, sensors, sensing technologies, crowd sourcing, crowd sensing, and development of the IoT and wireless sensor networks.



RABEB MIZOUNI (Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical and computer engineering from Concordia University, Montreal, Canada, in 2002 and 2007, respectively. She is currently an Associate Professor in electrical and computer engineering with the Khalifa University of Science and Technology. Her current research interests include the deployment of context aware mobile applications, crowd sensing, software product line, and cloud computing.



HADI OTROK (Senior Member, IEEE) received the Ph.D. degree in ECE from Concordia University. He holds a Professor position at the Department of ECE, Khalifa University of Science and Technology, an affiliate Associate Professor at the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada, and an affiliate Associate Professor at the Electrical Department, Ecole de Technologie Supérieure (ETS), Montreal. He is an Associate Editor at: *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, *Ad-hoc Networks* (Elsevier), and *IEEE Network*. His research interests include the domain of blockchain, reinforcement learning, crowd sensing and sourcing, ad-hoc networks, and cloud security. He was the Co-Chair of several committees at various IEEE conferences.

...