# Privacy Meets Regulations: Shaping the Future of Work

Mohammad Javad Amiri
Stony Brook University
amiri@cs.stonybrook.edu

Tristan Allard
University of Rennes, CNRS, IRISA
tristan.allard@irisa.fr

Boon Thau Loo
University of Pennsylvania
boonloo@seas.upenn.edu

Divyakant Agrawal
University of California Santa Barbara
agrawal@cs.ucsb.edu

Amr El Abbadi
University of California Santa Barbara
amr@cs.ucsb.edu

## ABSTRACT

Crowdworking platforms facilitate diverse workers in executing tasks for various requesters, contributing to the growth of the gig economy and the emergence of competing and complementary independent platforms. This has led to the development of multi-platform crowdworking systems, where workers and requesters often engage with multiple platforms. Recently, there has been an increasing interest among governmental, legal, and social institutions in enforcing regulations, such as minimum and maximum work hours, on these platforms. Consequently, collaboration among platforms within multi-platform systems is essential to enforce these cross-platform regulations effectively. However, while such collaboration necessitates the transparent sharing of information regarding tasks and participants, it is crucial to preserve the privacy of all involved participants. This paper outlines a vision for regulating, preserving privacy, and structuring future multi-platform crowdworking environments. We propose a potential instance of a multi-platform crowdworking system capable of enforcing a significant subset of practical global regulations across distributed independent platforms while preserving privacy through the use of lightweight anonymous tokens and fault-tolerant protocols.

## 1 INTRODUCTION

The rise of the "gig" or *platform economy* [21, 32] is reshaping work all around the world and has given rise to independent platforms that provide competing and complementary services. In general, crowdsourcing platforms dedicated to work (also called *crowdworking platforms* [17]) are online intermediaries between *requesters* and *workers*, where requesters propose *tasks* while *workers* propose skills and time. By providing requesters (workers) 24/7 access to a worldwide workforce (worldwide task market), crowdworking platforms have grown in numbers, diversity, and adoption. Today, crowdworkers come from all over the world and work on several, possibly competing, platforms [17]. The use of crowdworking platforms is expected to continue growing [42], and in fact, they are envisioned as key components of the future of work [7, 18, 29].

Crowdworking platforms, however, challenge national boundaries and weaken the formal relationships between the platforms, workers and task requesters. Guaranteeing the compliance of crowdworking platforms with national or regional labor laws is hard [5, 42] despite the critical need to rigorously regulate the gig workspace. For example, the total work hours of a worker per week may not exceed 40 hours to follow Fair Labor Standards Act (FLSA) [41]. In California, Assembly Bill 5 (AB5) law [2] entitled workers to greater labor protections, such as minimum wage laws, sick leave, and workers' compensation benefits. *AB5* was then overturned by *California Proposition 22*, which imposes its own set of regulations on minimal hours worked for health benefits.

The global regulation of work hours represents the *minimal* and *maximal* number of hours that participants, i.e., workers, requesters, and platforms, can spend on crowdworking platforms. While legal tools are currently being investigated [39, 42], there is an urgent need for technological tools to allow official institutions to enforce regulations. But as these regulatory mechanisms go forward, it is important to consider the differences between the gig economy's privacy and surveillance issues and those of a traditional service economy. In Uber, for example, the automatic aggregation of driver data even when the driver is not compensated, but driving on dead miles without carrying a fare, is still generating useful data for Uber, which is transmitted to the platform where the company analyzes traffic patterns and improves its algorithm [43].

The privacy issues become even more complex, as workers and requesters with specific tasks may need to work for or use the services of multiple platforms. In such an environment, the legal regulations need to be enforced over multiple possibly competing crowdworking platforms. Some platforms have already implemented self-defined *local regulations*. For example, Uber [4] and Lyft [3] force drivers to rest at least 6 hours for every 12 hours in driver mode, or Wirk.io prevents micro-workers from earning more than €3000 per year [1]. However, since workers and requesters can simply switch platforms when a local limit is reached, no global cross-platform regulation can be enforced. Moreover, participants in a crowdworking task may also behave maliciously [15], e.g., violate participants' privacy or the regulations.

Most current crowdworking platforms are independent of each other. However, the emergence of complex tasks that may need multiple contributions from possibly different platforms, on one hand, and more importantly, the enforcement of legal regulations, on the other hand, highlights the need for collaboration between crowdworking platforms, resulting in *multi-platform crowdworking systems*. For example, many drivers work for both Uber and Lyft concurrently [6], while a requester may also request multiple rides from both Uber and Lyft in parallel. The observation holds for microtask platforms [17] as well, where a requester who has registered on *Amazon Mechanical Turk* and *Prolific* might need hundreds of contributions for a single microtask and accept these contributions from workers regardless of the platforms the microtasks are performed on. Since workers from different platforms might want to perform contributions, the system needs to establish consensus among different microtask platforms to assign workers and provide the specified number of contributions while ensuring minimal and maximum hourly regulations on participants without revealing any private information to the competing platforms.

The emergence of Large language models (LLMs) also has a significant impact on crowdworking environments. On one hand, LLMs present exciting opportunities to enhance efficiency (e.g., automating repetitive and mundane tasks), accessibility (e.g., translating tasks and instructions), and productivity (e.g., dynamic matching between requesters and workers) in crowdworking environments. Conversely, LLMs raise significant challenges related to quality assurance (e.g., the quality and accuracy of generated content), ethical considerations (e.g., concerns over authorship, intellectual property rights, and the potential for generating misleading or harmful content), biases (e.g., resulting from inherit biases present in their training data), and data privacy (e.g., violating user data confidentiality) that need to be addressed.

In this paper, we present a *design space* for future of work multi-platform crowdworking environments. Our proposed design space consists of four main dimensions: *regulations*, *privacy*, *architecture*, and *content*.

First, a multi-platform crowdworking system must clearly define the *types of regulations* supported in terms of the *complexity* of the regulation (e.g., a simple or a chain of interactions) as well as the *aggregate* requirements of the regulation (e.g., no aggregation or SUM aggregations). For example, California Proposition 22, which states "if a driver works at least 25 hours per week, companies (i.e., platforms) require providing healthcare subsidies ...", is a simple, SUM-aggregate regulation.

Second, the threat model (e.g., *honest-but-curious*, *covert*, *fully malicious*) as well as the privacy guarantees provided to each entity, must be clearly specified.

Third, from an architecture design point of view, any multi-platform crowdworking environment consists of two critical components: *regulation management* that models and enforces regulations and *global state management* that manages the global states of all participants as well as tasks. Each of these components can be implemented in a centralized or a decentralized approach. Moreover, the proposed architecture needs to be scalable to manage ever-growing volumes of tasks in crowdworking environments.

Finally, a multi-platform crowdworking environment should be able to check and validate the content of the requested tasks and corresponding contributions against rules and regulations, e.g., detecting duplicate or incorrect contributions.

We then propose a potential point in the design space of multi-platform crowdworking environments. Our proposed system uses *lightweight* and *anonymous tokens* to enforce *privacy*, while *transparency* is achieved using fault-tolerant *distributed ledgers* shared among multiple platforms. The complexity of the conjunction of the required properties makes the problem non-trivial.

First, the regulations must be expressed in a *simple* and *unambiguous* manner. Second, while enforcing regulations over multiple crowdworking platforms requires the global state of the system to be *transparent*, the *privacy* of participants needs to be preserved,

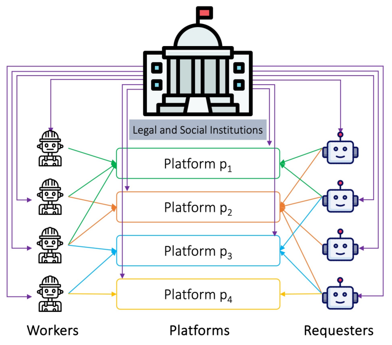

**Figure 1: A crowdworking infrastructure**

hence *transparency* needs to be reconciled with *privacy*. Third, the decentralized management of the global state among a distributed set of crowdworking platforms requires distributed consensus protocols and scalability techniques. Finally, we need to ensure that the content of tasks and corresponding contributions is valid.

## 2 DESIGN SPACE OF REGULATION SYSTEMS

Designing a regulation system for crowdworking platforms requires four essential decisions. First, the system must define the *types of regulations* it will implement, considering the various regulations relevant to different working environments. Second, given that the crowdworking ecosystem consists of distributed and potentially untrustworthy entities (including platforms, workers, and requesters), the system needs to clearly outline the *privacy guarantees* provided to each entity, along with any computational guarantees. Third, the system must specify the *validation mechanisms* that will be used to verify the content of tasks and contributions. Finally, the inherently distributed nature of crowdworking demands a range of *architectural choices*, which can span from fully decentralized to more traditional centralized frameworks.

### 2.1 Crowdworking Environment

A *crowdworking environment* [30, 34–36, 46–48] consists of a set of workers $\mathcal{W}$ interacting with a set of requesters $\mathcal{R}$ through a set of competing platforms $\mathcal{P}$. We refer to the workers, platforms, and requesters of a crowdworking environment as *participants*. Each worker $w \in \mathcal{W}$ (1) registers to one or more platforms $\mathcal{P}_w \subset \mathcal{P}$ according to her preferences and, through the latter, (2) accesses the set of tasks available on $\mathcal{P}_w$, (3) submits each *contribution* to the platform $p \in \mathcal{P}_w$ she elects, and (4) obtains a *reward* for her work. On the other hand, each requester $r \in \mathcal{R}$ similarly (1) registers to one or more platforms $\mathcal{P}_r \subset \mathcal{P}$, (2) issues a *submission* which contains her tasks $\mathcal{T}_r$ to one or more platforms $p \in \mathcal{P}_r$, (3) receives the contributions of each worker $w$ registered to $\mathcal{P}_r \cap \mathcal{P}_w$ having elected a task $t \in \mathcal{T}_r$, and (4) launches the distribution of rewards. Platforms are thus in charge of facilitating the interactions between workers and requesters. A *crowdworking process* $\pi$ connects three participants – a workers $w$, a platform $p$, and a requester $r$ – and aims to facilitate the execution of a task $t \in \mathcal{T}_r$ through platform $p$ via a worker $w$. For simplicity and without loss of generality, we assume that each process corresponds to a time unit of work (e.g., 1 hour). Figure 1 shows a crowdworking infrastructure with four platforms, four workers, and four requesters.

## 2.2 Types of Regulation

The space of possible regulations can be structured based on two orthogonal dimensions: the *complexity* of the regulation, e.g., constraints on a single process versus constraints that apply to multiple processes that are transitively related, and the *aggregate* nature of the regulation, e.g., no aggregation versus restrictions involving SUM aggregate.

In order to give a clear semantics to each dimension and to rigorously map regulations to points in this space, we express regulations by SQL constraints over a *universal virtual table* storing information about all crowdworking processes having been performed. We denote this table by U-TABLE and emphasize that it is *virtual* (we use it only for clarifying the various types of regulations; it is never instantiated). The attributes of U-TABLE refer to metadata about the interactions (i.e., at least the worker, requester, and platform involved in a process, and also possibly additional metadata such as begin and end timestamps) and information about the contents (e.g., the time estimate for the task[1] - the TIMECOST attribute below -, the proposed wage, the worker's contribution). For simplicity, we focus below on the attributes of the U-TABLE relevant to our illustrative examples: (1) TS_BEGIN, TS_END, WORKER, PLATFORM, and REQUESTER, and (2) TIMECOST, WAGE, and CONTRIBUTION. Finally, a regulation is simply a Boolean SQL expression nested within the usual CHECK clause: `ALTER TABLE U ADD CONSTRAINT r CHECK ( ... )`.

The complexity and aggregate dimensions of a regulation can both be deduced from its SQL expression. The complexity is given by the presence of *join* operations while the aggregate dimension is given by the presence of *aggregate* function(s), possibly with GROUP BY and HAVING clause(s). For simplicity, we consider a coarse-grained characterization of these two dimensions. A regulation is `simple` if there is no join and `complex` otherwise. A regulation is `row-only` if it does not involve any aggregate function, `aggregate-only` if it involves only comparison(s) over aggregate(s), and `mixed` if it involves comparisons over rows and aggregates.

We illustrate the possible types of regulations based on simple examples extracted from real-life crowdworking regulations or from real-life proposals of regulation. First, we consider a regulation $r_1$ requiring the wage proposed by each task to be at least a given amount $\theta$. It illustrates the (`simple`, `row-only`) type of regulation.

```
ALTER TABLE U-TABLE ADD CONSTRAINT r1 CHECK (
  NOT EXISTS (
    SELECT * FROM U
    WHERE TIMECOST ≤  θ
  ) );
```

Second, we consider a regulation $r_2$ requiring each worker to work at most a given number of time units $\theta$ per time period $\rho$. It illustrates the (`simple`, `mixed` with SUM-aggregate) type of regulation. It is similar to the regulation followed by the `wirk.io` platform that limits the crowdworking gains of any worker on the platform to 3000 euros per year. The following SQL constraint expresses $r_2$, assuming that `current_time()` gives the current time in the same unit as the period $\rho$.

```
ALTER TABLE U-TABLE ADD CONSTRAINT r2 CHECK (
  NOT EXISTS (
    SELECT * FROM U
    WHERE WORKER=w AND current_time()-TS_BEGIN ≤  ρ
    GROUP BY WORKER
    HAVING SUM(TIMECOST) ≥  θ
  ) );
```

Although most regulations must always hold, e.g., a lower than constraint on a (`simple`, SUM-aggregate) regulation, a (`complex`, `row-only`) regulation, some regulations, inherently, *cannot* always hold. Similar to `deferred` SQL constraints, they must only hold after a given time period. For example, a periodic greater than constraint on a (`simple`, SUM-aggregate) regulation cannot hold initially, but must hold at the end of a given period (e.g., a constraint that requires a worker to work at least 25 hours per week to qualify for healthcare subsidies). We call *enforceable* the regulations that must always hold and *verifiable* the regulations that eventually hold. The verifiable/enforceable property of a regulation is only due to its nature, not to its implementation (but it impacts it directly). Future crowdworking regulation systems need to determine the enforceable/verifiable properties of the regulations they support.

## 2.3 Threat Model and Privacy Model

Crowdworking environments are open environments that connect possibly adversarial participants. Any regulation system must clearly specify both the threat model (e.g., *honest-but-curious*, *covert*, *fully malicious*) and the privacy model. While the threat model only depends on the underlying system, the privacy model can be based on a common formal requirement parameterized for each system by the leaks it tolerates. The possible leakage ranges from no information leaked to any participant (similar to usual *secure multi-party computation* algorithms) to full disclosure (of all the information discussed above) to all participants (e.g., in current crowdworking platforms, the underlying system often requires full disclosure to the platform). The disclosures tolerated by future regulation systems will fall within this range.

We formalize below a common privacy model based on the well-known simulatability paradigm often used by secure multi-party computation algorithms. The proposed model guarantees that nothing leaks, with computational guarantees[2], except the *pluggable* system-dependent tolerated *disclosures*.

Consider a crowdworking process $\pi$ between worker $w$, platform $p$, and requester $r$ for solving a task $t$. The task may have been sent to several platforms and, depending on the underlying crowdworking platforms, might have been accessed by several workers before being picked and solved. The information generated by the execution of $\pi$ consists, similarly to the information captured by the U-TABLE, of information about interactions (e.g., at least $w$, $r$, and

[1]Future regulation systems will need to design technical means to guarantee the reliability of the time estimates for tasks (e.g., privacy-preserving feedback systems from workers, automatic time estimation by analyzing task descriptions).

[2]The majority of data protection techniques used in real-life are based on encryption schemes that provide guarantees against computationally-bounded adversaries, but the model can be easily adapted to information-theoretic attackers.

$p$, the participants directly involved in $\pi$) and information about content (e.g., description of $t$, contribution, proposed wage). We propose to define the sets of disclosure according to the involvement of a participant in $\pi$. Indeed, the platforms not involved in $\pi$ (i.e., different from $p$) but that received $t$ may need to learn that $t$ has been completed (e.g., to manage their local copy of the task). Similarly, workers who are not involved in a task $t$ might still need to know that it has been executed, while potentially preserving the privacy of worker $w$ who executed the task. The three sets of disclosures defined below cover all these cases[3]. Future regulation systems have to specify clearly the content of each disclosure set.

- Disclosures to the participants that are **not involved** in $\pi$ and that have **not received** task $t$ from requester $r$: $\delta^{\pi}_{\neg R \neg I}$
- Disclosures to the platforms and workers that have **received** the task $t$ from $r$ but that are **not involved** in $\pi$: $\delta^{\pi}_{R \neg I}$
- Disclosures to the participants that are directly **involved** in $\pi$ (and have thus **received** task $t$): $\delta^{\pi}_{RI}$

Let $\Pi$ be a set of crowdworking processes executed by $\varsigma$ a regulation system over a set of participants. We say that $\varsigma$ is $\delta^{\Pi}$*-Private* if, for all $\pi \in \Pi$, for all computationally-bounded adversaries $\mathsf{A}$, the sets of disclosures $(\delta^{\pi}_{\neg R \neg I}, \delta^{\pi}_{R \neg I}, \delta^{\pi}_{RI})$, assuming arbitrary background knowledge $\chi \in \{0,1\}^*$, the distribution representing the adversarial knowledge over the input dataset in the real setting is *computationally indistinguishable* from the distribution representing the adversarial knowledge in an ideal setting in which a trusted third party $\mathsf{cp}$ executes the crowdworking process $\pi$ of $\varsigma$:

$$\mathsf{REAL}_{\varsigma, \mathsf{A}(\chi, \delta^{\pi}_{i})}(\mathcal{W}, \mathcal{P}, \mathcal{R}, \mathcal{T}) \overset{c}{\equiv} \mathsf{IDEAL}_{\mathsf{cp}, \mathsf{A}(\chi, \delta^{\pi}_{i})}(\mathcal{W}, \mathcal{P}, \mathcal{R}, \mathcal{T})$$

where $i \in \{\neg R \neg I, R \neg I, RI\}$, and $\mathsf{REAL}$ denotes the adversarial knowledge in the real setting and $\mathsf{IDEAL}$ its counterpart in the ideal setting.

## 2.4 Content

Crowdsourcing platforms employ various techniques, such as quality control mechanisms, reputation-based systems, and automated checks, to monitor and validate worker contributions, ensuring task integrity and reliability. Quality control often involves using predetermined "gold standard" tasks with known answers to evaluate accuracy. Additionally, having multiple workers complete the same task allows for comparison of results to identify outliers and reach a consensus on correct answers. In reputation systems, workers receive scores based on past performance, where lower scores may result in increased scrutiny or reduced task assignments. Moreover, many platforms implement algorithms and automated tools to assess submissions for adherence to predefined rules and regulations, including consistency, grammar, and relevance.

The validity of content can also be governed by SQL constraints, as discussed in Section 2.2. For example, regulation $r_3$ prohibits any worker from submitting similar contributions to the same requester, even across different platforms. The `sim` function measures the similarity of contributions, with threshold $\theta$ defining the similarity limit. This regulation exemplifies a (`complex`, `row-only`) type.

```
ALTER TABLE U-TABLE ADD CONSTRAINT r3 CHECK (
 NOT EXISTS (
   SELECT *
   FROM U U1 JOIN U U2 ON
       U1.WORKER=U2.WORKER
       AND U1.REQUESTER=U2.REQUESTER
       AND sim(U1.CONTRIBUTION, U2.CONTRIBUTION) ≥ θ
 ) );
```

[3]This can be extended by tuning the disclosure sets (e.g., by distinguishing the requesters from the platforms in the set of involved participants).

## 2.5 Architecture

The architecture of a multi-platform crowdworking system comprises two primary components: *regulation management* and *global state management*. Regulation management is responsible for modeling the regulations governing participant interactions and ensuring compliance among all participants. In contrast, global state management maintains the overarching states of the system, encompassing all relevant information about participants and tasks. For the implementation of these components, a centralized or decentralized approach can be utilized. A centralized approach facilitates rapid prototyping but necessitates additional technologies to ensure fault tolerance, privacy, and trustworthiness. Conversely, a decentralized approach aligns well with diverse organizational entities but introduces increased overhead and more complex communication protocols.

Additionally, each platform generates substantial data, and a multi-platform crowdworking environment must efficiently process numerous transactions within or across platforms while maintaining scalability. Sharding, a proven technique for enhancing scalability, involves partitioning platform data into multiple data shards [16, 19, 22, 24, 28, 31, 44, 45]. Sharding has been widely studied under Byzantine failures [13, 40], especially in the domain of blockchains [8, 9, 11, 13, 14, 23, 26, 27, 33, 38, 49]. However, applying sharding in multi-platform environments poses challenges, particularly regarding data privacy. Such a system must accommodate various transaction types that access one or multiple data shards across platforms.

Another architectural challenge revolves around the verifiability and enforceability of regulations. While enforceable regulations can be upheld within the multi-platform system, verifiable regulations may be relevant to external entities such as legal courts or insurance companies. In these cases, the system must provide credible evidence to demonstrate compliance with regulations, facilitating the resolution of any disputes that may arise.

## 3 AN INSTANCE OF REGULATION SYSTEMS

Inspired by Separ [10], we present a possible instance in the design space of crowdworking systems to enforce regulations on a set of distributed platforms in a privacy-preserving manner. We propose a *two-level* solution consisting of a *privacy-preserving token-based system* (i.e., the *application level*) on top of a *distributed ledger* shared across platforms (i.e., the *infrastructure level*). First, at the application level, global regulations are modeled using *lightweight* and *anonymous tokens* distributed to workers, platforms, and requesters. The information shared among participants is limited to the minimum necessary for performing the tasks against adversarial participants acting as covert adversaries. Second, at the infrastructure level, the immutable ledger provides transparency across platforms.

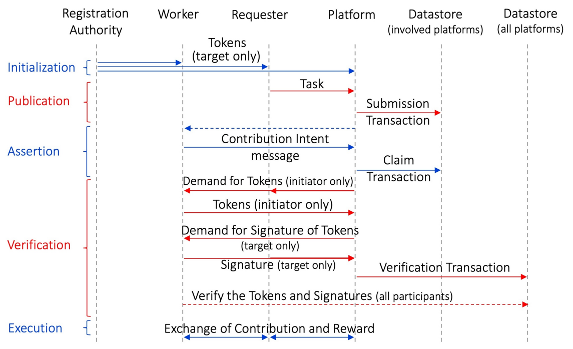
Registration Authority
Worker
Requester
Platform
Datastore (involved platforms)
Datastore (all platforms)
Initialization
Publication
Assertion
Verification
Execution
Tokens (target only)
Task
Submission Transaction
Contribution Intent message
Claim Transaction
Demand for Tokens (initiator only)
Tokens (initiator only)
Demand for Signature of Tokens (target only)
Signature (target only)
Verification Transaction
Verify the Tokens and Signatures (all participants)
Exchange of Contribution and Reward

**Figure 2: Sequence chart (references to targets include all participants for $v$-tokens)**

Nonetheless, for the sake of privacy and performance, the ledger is *not maintained* by any platform, and each platform maintains only a view of the ledger. Consensus protocols are used to deal with the concurrency issues inherent to a multi-platform context.

Given the large variety of regulations that apply to working environments, any given regulation system must clearly define the *types of regulations* supported in terms of the *complexity* of the regulation (e.g., a simple or a chain of interactions) as well as the *aggregate* requirements of the regulation (e.g., no aggregation or SUM aggregations). We focus only on managing lower and upper bounds on aggregate-based regulations. Although most regulations must always hold, some regulations cannot be held initially but must be held at the end of a given period. For example, CA Proposition 22 requires workers to work at least 25 hours per week to qualify for healthcare subsidies.

Enforceable are the regulations that must always hold and verifiable are the regulations that eventually hold. The proposed system is able to implement a small subset of both enforceable and verifiable regulations by managing two *budgets* per participant while guaranteeing both privacy and correctness. Budgets are modeled using *lightweight* and *anonymous tokens* distributed to workers, platforms, and requesters. The overall system is conceptually simple and only relies on correctly using individual and group signatures. Participants in a crowdworking environment might be located in different states or even countries where different, possibly conflicting regulations exist. In this case, crowdworking environments rely on official institutions to define the regulations that need to be enforced, e.g., a gig worker might need to follow the regulations of their location, the regulations of the platform's location, or possibly a new set of regulations. The question is how to enforce different possible decisions made by official institutions regarding conflicting regulations.

Our initial design uses a centralized regulation management component called the registration authority (RA) to bootstrap the system and its participants and refresh their budgets periodically, but the RA is not involved in the continuous execution of crowdworking processes and their regulations. The registration authority, on the one hand, needs to be trusted by all participants, and on the other hand, might be under high load. To deal with this challenge, global regulations can be enforced in a decentralized manner where all participants are involved in verifying the global task and contribution regulations. To achieve verifiability, we can build our system using zero-knowledge proofs [37, 50], especially, zk-SNARKs [25] to address the integrity of constraints. It basically requires the untrusted entity (e.g., crowdworking platform) to provide verifiable proofs that they actually perform the correct actions they claim.

To regulate crowdworking processes, the token-based system is defined by five functions: `GENERATE` for initializing the budgets with the correct number of tokens and refilling them, `SPEND` for spending portions of the budgets, `PROVE` for providing proofs for verifying verifiable regulations (e.g., to a third party), `CHECK` for checking whether a given spending is allowed or not, and `ALERT` for reporting dubious spending. Since the execution of these functions changes the global state of the system, the data involved in the execution (e.g., tasks, tokens, signatures) must be appended to the immutable ledger of the platforms.

The processing of a crowdworking task involves five main phases, as shown in Figure 2. Initialization where the registration authority provides all parties with their keys and tokens. Publication where the requesters create and send their tasks to platforms, and platforms append a submission transaction to the ledger. Assertion where the worker sends a *contribution intent* message to the platform, and the platform appends a claim transaction to the ledger. Verification where the platform asks the corresponding requester and worker to send the required tokens and signatures through the `SPEND` function, and the platform appends $e$-tokens and their signatures to the ledger through verification transactions, and finally Execution where the actual contribution can be delivered to the requester and the reward to the worker.

Efficient data management across multiple platforms necessitates a data model capable of storing local, cross-platform, and global data while preserving privacy [12]. Furthermore, a platform may participate in various crowdworking workflows involving different sets of entities. We construct a hierarchical data model comprising a series of *data collections*, each facilitating transactions within its respective context. The root data collection stores global records from executing verification transactions, replicated across all platforms. Local data collections comprise records of internal tasks specific to each platform, while intermediate data collections manage records of cross-platform tasks.

Figure 3(a) illustrates a data model for a crowdworking scenario involving platforms $A$, $B$, $C$, and $D$, with all global records stored in the aggregate data collection $d_{ABCD}$. As depicted in Figure 3(b), platform $A$ maintains its local data collection $d_A$, the root data collection $d_{ABCD}$, and relevant intermediate data collections such as $d_{AB}$ and $d_{ACD}$. Figure 3(c) presents two distinct data models for different crowdworking workflows where platforms $K$, $L$, and $M$ are involved in the first and $L$, $M$, and $N$ are involved in the second workflow.

Within each platform, a set of replicated nodes stores copies of the platform's data to provide fault tolerance. Nodes of the same or different platforms need to establish consensus on a unique order in which entries are appended to the ledger. To establish consensus among the nodes and since different entities do not trust each other, Byzantine fault-tolerant protocols, e.g., PBFT [20], have been used. Completion of a crowdworking task requires a single submission

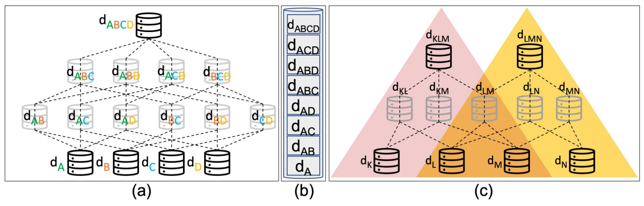

**Figure 3: (a) A data model for a 4-platform workflow, (b) data maintained by $A$, (c) data models for 3-platform workflows.**

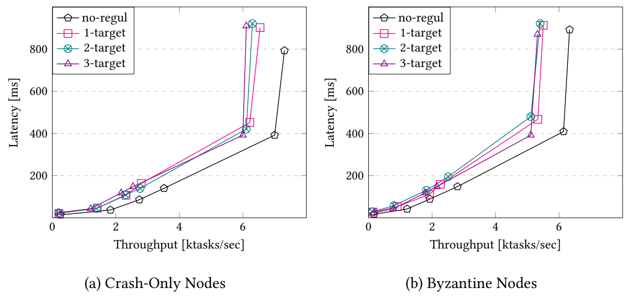

**Figure 4: Class of regulations**

transaction, one or more claim transactions (depending on the requested number of contributions), and a verification transaction. For an internal task of a platform, transactions are replicated only on the platform's nodes using *local consensus*. For a cross-platform task, on the other hand, transactions need to be replicated on every node of all involved platforms using *cross-platform consensus*. Finally, some transactions need to be appended to the ledger of all platforms. Therefore, all nodes of *every* platform participate in a *global consensus* protocol.

## 4 PRELIMINARY EVALUATION

We have implemented an early proof-of-concept prototype of our system based on Separ (a simple centralized token-based RA with a set of 4 platforms) and experimentally evaluated its performance and scalability. We consider a setting with 4 platforms where requesters and workers are randomly registered to one or more platforms. Submitted tasks are randomly assigned to idle workers (to avoid any delay). We mainly focus on the overhead of preserving privacy when satisfying enforceable regulations. The experiments were conducted on the Amazon EC2 platform. Each VM is a c4.2xlarge instance with 8 vCPUs and 15GB RAM, Intel Xeon E5-2666 v3 processor clocked at 3.50 GHz.

### 4.1 Token Generation

We first measure the performance of token generation (performed by RA). We consider different classes of regulations, i.e., single-target (e.g., $((w, *, *), <, \theta)$), two-target (e.g., $((*, p, r), <, \theta)$), and three-target (e.g., $((w, p, r), <, \theta)$) enforceable regulations. Our experiments show that our system is able to generate tokens in linear time. In our experiments, each token was generated in $0.07ms$, hence, generating 1 million tokens in $\sim$ 76 seconds. This demonstrates the scalability of the token generation, especially since token generation is executed periodically, e.g., every week or every month. Note that we use a single machine to generate tokens. However, tokens related to different regulations can be generated in parallel, e.g., with 10 machines, generating 1 million tokens takes $\sim$ 7.6 seconds. Furthermore, the class of regulations (i.e., the number of targets) does not affect the performance, i.e., the throughput and latency of token generation are constant in terms of the number of targets. It should, however, be noted that regulations with more targets requires more tokens to be generated, e.g., $((w, *, *), <, \theta)$ requires $|\mathcal{W}| * \theta$ tokens, whereas $((w, p, r), <, \theta)$ requires $|\mathcal{W}| * |\mathcal{P}| * |\mathcal{R}| * \theta$ tokens to be generated.

### 4.2 Classes of Regulations

Next, we measure the overhead of privacy-preserving techniques. We consider the basic scenario with no regulation (i.e., no need to exchange and validate tokens and signatures) and compare it with three different scenarios where each task has to satisfy a single target, a two-target, and a three-target regulation. The system consists of four platforms, and the workload includes 90% internal and 10% cross-platform tasks, where two (randomly chosen) platforms are involved in submission and claim transactions of cross-platform tasks. Note that all platforms are still involved in the verification transaction of each task. We also assume that completing each task requires a single contribution, i.e., claim transaction (and obviously a submission and a verification transaction).

When nodes follow the crash failure model, and the system includes no regulations, as shown in Figure 4(a), the system processes 7000 tasks with 390 ms latency (the penultimate point). Adding regulations results in more communication between participants to exchange tokens and signatures. However, the system still processes 6200 tasks with 450 ms latency. In fact, all used privacy-preserving techniques result in only 11% and 15% overhead in terms of throughput and latency, respectively. Moreover, the class of regulations does not significantly affect the performance. This is expected because more targets result in only increasing the number of (parallel) tokens and signature exchanges while the number of communication phases is not affected. Similarly, in the presence of Byzantine nodes and as shown in Figure 4(b), the system processes 6140 tasks with 409 ms latency with no regulations and 5331 tasks (13% overhead) with 467 ms (14% overhead) latency with single-target regulations. As before, the class of regulations does not affect the performance. Increasing the number of regulations still leads to similar behavior: while it may increase tokens, participants, and signatures, it does not impact the consensus and other communication phases.

## 5 CONCLUSION

In this paper, we outline a framework for the future of work multi-platform crowdworking environments, encompassing four key dimensions: regulations, security, content, and architecture. We design a system to enforce global regulations within these environments while preserving participant privacy. Our system enables official institutions to articulate global regulations in unambiguous language, ensures compliance with these regulations by design, and allows participants to verify their involvement in crowdworking tasks to external entities, all while maintaining privacy.

## REFERENCES

[1] [n. d.]. 50,000 people in France. https://www.wirk.io/50k-freelances-en-france.

[2] [n. d.]. Assembly Bill No. 5. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201920200AB5.

[3] [n. d.]. Introducing a new feature: Driving Hours Limit. https://help.lyft.com/hc/articles/115012926787-Taking-breaks-and-time-limits.

[4] [n. d.]. Taking breaks and time limits in driver mode. https://www.uber.com/en-ZA/blog/driving-hours-limit.

[5] [n. d.]. the `Otey V Crowdflower` class action against a famous microtask platform for *"substandard wages and oppressive working hours"*. casetext.com/case/otey-v-crowdflower-1.

[6] 2020. Make More Money by Driving for Uber and Lyft at the Same Time. https://rideshareapps.com/drive-for-uber-and-lyft-at-the-same-time/.

[7] Sihem Amer-Yahia, Senjuti Basu Roy, Lei Chen, Atsuyuki Morishima, James Abello Monedero, Pierre Bourhis, François Charoy, Marina Danilevsky, Gautam Das, Gianluca Demartini, et al. 2020. Making AI machines work for humans in FoW. *ACM Sigmod Record* 49, 2 (2020), 30–35.

[8] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. 2019. On Sharding Permissioned Blockchains. In *Int. Conf. on Blockchain*. IEEE, 282–285.

[9] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. 2021. SharPer: Sharding Permissioned Blockchains Over Network Clusters. In *SIGMOD Int. Conf. on Management of Data*. ACM, 76–88.

[10] Mohammad Javad Amiri, Joris Duguépéroux, Tristan Allard, Divyakant Agrawal, and Amr El Abbadi. 2021. SEPAR: Towards Regulating Future of Work Multi-Platform Crowdworking Environments with Privacy Guarantees. In *The Web Conf. (WWW)*. 1891–1903.

[11] Mohammad Javad Amiri, Ziliang Lai, Liana Patel, Boon Thau Loo, Eric Lo, and Wenchao Zhou. 2023. Saguaro: An Edge Computing-Enabled Hierarchical Permissioned Blockchain. In *Int. Conf. on Data Engineering (ICDE)*. IEEE, 259–272.

[12] Mohammad Javad Amiri, Boon Thau Loo, Divyakant Agrawal, and Amr El Abbadi. 2022. Qanaat: A Scalable Multi-Enterprise Permissioned Blockchain System with Confidentiality Guarantees. *Proc. of the VLDB Endowment* 15, 11 (2022), 2839–2852.

[13] Mohammad Javad Amiri, Sujaya Maiyya, Daniel Shu, Divyakant Agrawal, and Amr El Abbadi. 2023. Ziziphus: Scalable Data Management Across Byzantine Edge Servers. In *Int. Conf. on Data Engineering (ICDE)*. IEEE, 490–502.

[14] Elli Androulaki, Christian Cachin, Angelo De Caro, and Eleftherios Kokoris-Kogias. 2018. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In *European Symposium on Research in Computer Security (ESORICS)*. Springer, 111–131.

[15] Yonatan Aumann and Yehuda Lindell. 2007. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography Conf. (TCC)*. Springer, 137–156.

[16] Jason Baker, Chris Bond, James C Corbett, JJ Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, and Vadim Yushprakh. 2011. Megastore: Providing scalable, highly available storage for interactive services. In *Conf. on Innovative Data Systems Research (CIDR)*.

[17] Janine Berg, Marianne Furrer, Ellie Harmon, U Rani, and M Six Silberman. 2018. *Digital labour platforms and the future of work: Towards decent work in the online world*. Int. Labour Office Geneva.

[18] Pierre Bourhis, Gianluca Demartini, Shady Elbassuoni, Emile Hoareau, and H Raghav Rao. 2019. Ethical challenges in the future of work. *Bulletin of the Technical Committee on Data Engineering* (2019).

[19] Nathan Bronson, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, and Harry Li. 2013. TAO: Facebook's Distributed Data Store for the Social Graph. In *Annual Technical Conf. (ATC)*. USENIX Association, 49–60.

[20] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine fault tolerance. In *Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 173–186.

[21] Julie E Cohen. 2017. Law for the platform economy. *UCDL Rev.* 51 (2017), 133.

[22] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, and Peter Hochschild. 2013. Spanner: Google's globally distributed database. *Transactions on Computer Systems (TOCS)* 31, 3 (2013), 8.

[23] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. 2019. Towards Scaling Blockchain Systems via Sharding. In *SIGMOD Int. Conf. on Management of Data*. ACM, 123–140.

[24] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: Amazon's highly available key-value store. *Operating Systems Review (OSR)* 41, 6 (2007), 205–220.

[25] Dario Fiore, Anca Nitulescu, and D. Pointcheval. 2020. Boosting Verifiable Computation on Encrypted Data. *IACR Cryptol. ePrint Arch.* 2020 (2020), 132.

[26] Ethan Frey and Christopher Goes. 2018. Cosmos Inter-Blockchain Communication (IBC) Protocol. https://cosmos.network.

[27] Danezis George and Sarah Meiklejohn. 2016. Centrally Banked Cryptocurrencies. In *Network and Distributed System Security Symposium (NDSS)*.

[28] Lisa Glendenning, Ivan Beschastnikh, Arvind Krishnamurthy, and Thomas Anderson. 2011. Scalable consistency in Scatter. In *Symposium on Operating Systems Principles (SOSP)*. ACM, 15–28.

[29] David Gross-Amblard, Atsuyuki Morishima, Saravanan Thirumuruganathan, Marion Tommasi, and Ko Yoshida. 2019. Platform design for crowdsourcing and future of work. *Bulletin of the Technical Committee on Data Engineering* 42, 4 (2019).

[30] Siyuan Han, Zihuan Xu, Yuxiang Zeng, and Lei Chen. 2019. Fluid: A blockchain based framework for crowdsourcing. In *SIGMOD Int. Conf. on Management of Data*. ACM, 1921–1924.

[31] Robert Kallman, Hideaki Kimura, Jonathan Natkins, Andrew Pavlo, Alexander Rasin, Stanley Zdonik, Evan PC Jones, Samuel Madden, Michael Stonebraker, and Yang Zhang. 2008. H-store: a high-performance, distributed main memory transaction processing system. *Proc. of the VLDB Endowment* 1, 2 (2008), 1496–1499.

[32] Martin Kenney and John Zysman. 2016. The rise of the platform economy. *Issues in science and technology* 32, 3 (2016), 61.

[33] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *Symposium on Security and Privacy (SP)*. IEEE, 583–598.

[34] An Liu, Zhi-Xu Li, Guan-Feng Liu, Kai Zheng, Min Zhang, Qing Li, and Xiangliang Zhang. 2017. Privacy-preserving task assignment in spatial crowdsourcing. (2017).

[35] An Liu, Weiqi Wang, Shuo Shang, Qing Li, and Xiangliang Zhang. 2018. Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. *GeoInformatica* 22, 2 (2018), 335–362.

[36] Bozhong Liu, Ling Chen, Xingquan Zhu, Ying Zhang, Chengqi Zhang, and Weidong Qiu. 2017. Protecting location privacy in spatial crowdsourcing using encrypted data. (2017), 478–481.

[37] Yuan Lu, Qiang Tang, and Guiling Wang. 2018. Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In *Int. Conf. on Distributed Computing Systems (ICDCS)*. IEEE, 853–865.

[38] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *SIGSAC Conf. on Computer and Communications Security (CCS)*. ACM, 17–30.

[39] Conseil national du numérique. 2020. Travail à l'ère des plateformes. Mise à jour requise. https://cnnumerique.fr/publication-du-rapport-travail-lere-des-plateformes-mise-jour-requise-en-presence-de-cedric-o (in French).

[40] Faisal Nawab and Mohammad Sadoghi. 2019. Blockplane: A global-scale byzantizing middleware. In *Int. Conf. on Data Engineering (ICDE)*. IEEE, 124–135.

[41] U.S. Department of Labor. [n. d.]. Wages and the Fair Labor Standards Act. https://www.dol.gov/agencies/whd/flsa.

[42] Global Commission on the Future of Work. 2019. *Work for a brighter future*. Technical Report ISBN 978-92-2-132796-7. Int. Labour Organization.

[43] Alex Rosenblat and Luke Stark. 2016. Algorithmic labor and information asymmetries: A case study of Uber's drivers. *International journal of communication* 10 (2016), 27.

[44] Rebecca Taft, Essam Mansour, Marco Serafini, Jennie Duggan, Aaron J Elmore, Ashraf Aboulnaga, Andrew Pavlo, and Michael Stonebraker. 2014. E-store: Fine-grained elastic partitioning for distributed transaction processing systems. *Proc. of the VLDB Endowment* 8, 3 (2014), 245–256.

[45] Alexander Thomson, Thaddeus Diamond, Shu-Chun Weng, Kun Ren, Philip Shao, and Daniel J Abadi. 2012. Calvin: fast distributed transactions for partitioned database systems. In *SIGMOD Int. Conf. on Management of Data*. ACM, 1–12.

[46] Hien To, Gabriel Ghinita, Liyue Fan, and Cyrus Shahabi. 2016. Differentially private location protection for worker datasets in spatial crowdsourcing. *Transactions on Mobile Computing* 16, 4 (2016), 934–949.

[47] Hien To, Cyrus Shahabi, and Li Xiong. 2018. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *Int. Conf. on Data Engineering (ICDE)*. IEEE, 833–844.

[48] Yansheng Wang, Tianshu Song, Qian Tao, Yuxiang Zeng, Zimu Zhou, Yi Xu, Yongxin Tong, and Lei Chen. 2019. Interaction Management in Crowdsourcing. *Data Engineering* (2019), 23.

[49] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling blockchain via full sharding. In *SIGSAC Conf. on Computer and Communications Security*. ACM, 931–948.

[50] Saide Zhu, Zhipeng Cai, Huafu Hu, Yingshu Li, and Wei Li. 2019. zkCrowd: a hybrid blockchain-based crowdsourcing platform. *Transactions on Industrial Informatics* (2019).