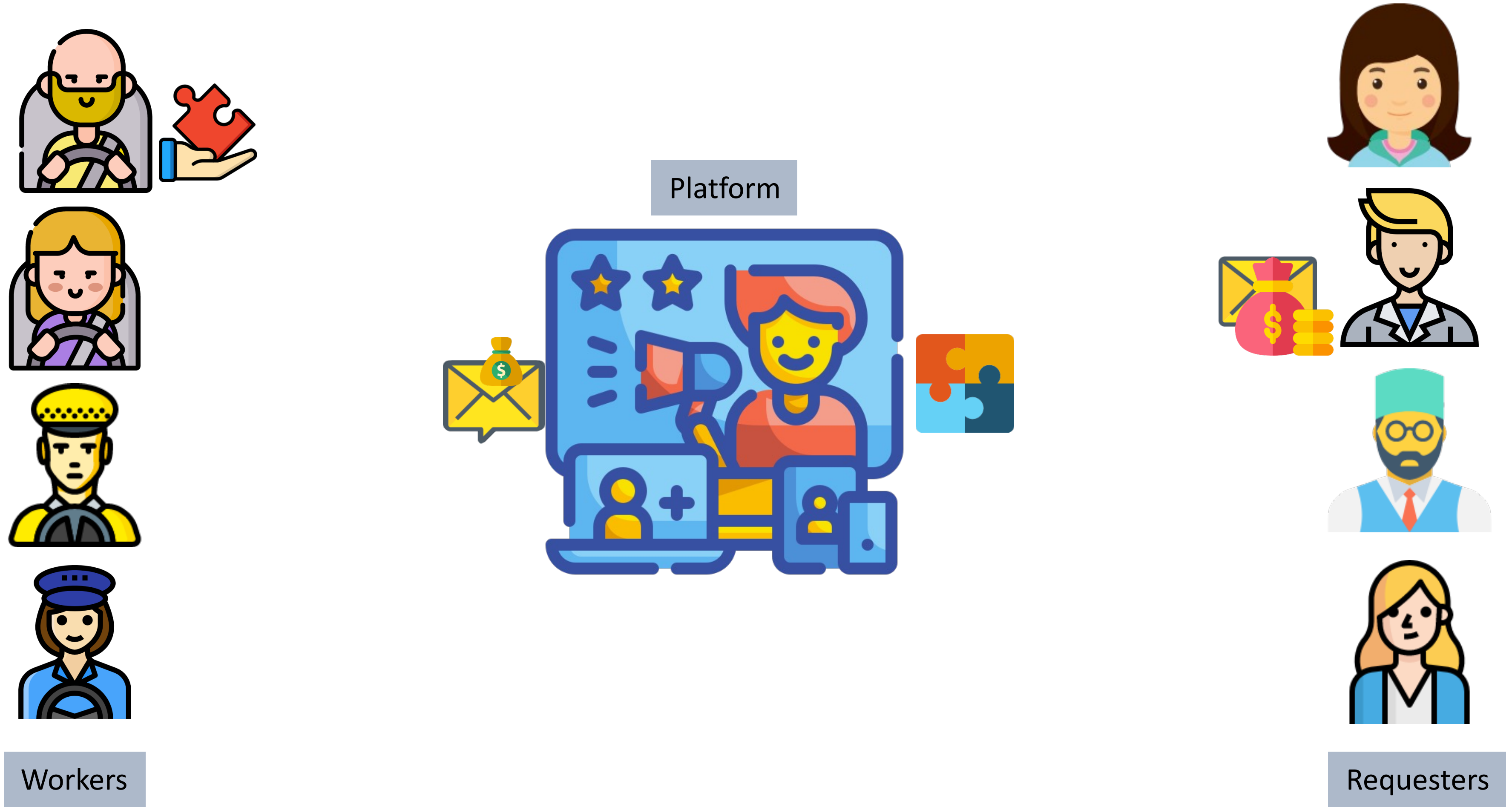
Separ: Towards Regulating Future of Work Multi-Platform Crowdworking Environments with Privacy Guarantees

Mohammad Javad Amiri[1], Joris Duguépéroux[2], Tristan Allard[2], Divyakant Agrawal[3], Amr El Abbadi[3]

[1]University of Pennsylvania, [2]Univ Rennes, CNRS, IRISA, [3]UC Santa Barbara

April 16, 2021

Crowdworking Environment



- Crowdworking platforms are online intermediaries between requesters and workers
- Envisioned as key technological components of the future of work

Guaranteeing the compliance of crowdworking platforms with regulations

"Whereas universal and lasting peace can be established only if it is based upon social justice;
. . . for example, by **the regulation of the hours of work** . . . "

preamble of the constitution of the International Labor Organization
[Commission on International Labor Legislation, 1919]

Figure: Members of the Commission on International Labor Legislation to the Paris Peace Conference (1919).

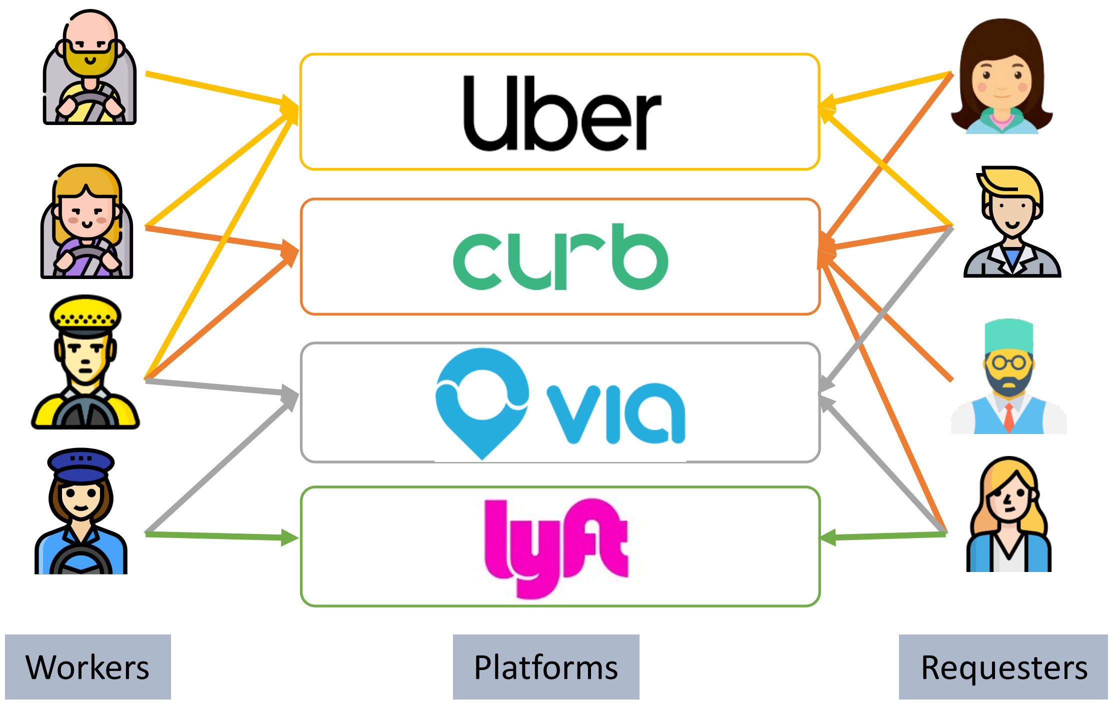
FLSA: Total work hours of a worker per week may not exceed 40 hours

In California, Assembly Bill 5 (AB5) entitles workers to greater labor protections, such as minimum wage laws, sick leave, and unemployment and workers' compensation benefits.

CA Proposition 22 imposes its set of regulations, e.g., requires a worker to work at least 25 hours per week to qualify for healthcare subsidies.

There is more than one platform …

- Workers often work on several platforms
- Requesters submit tasks on multiple platforms



Privacy Rights of Participants

- No participant obtains or infers any information beyond what is strictly needed
    - A driver who works for both Uber and Lyft, does not want either of them know that she works for the other.
- How to enforce regulations?
    - We need to reconcile transparency with privacy

Problems!

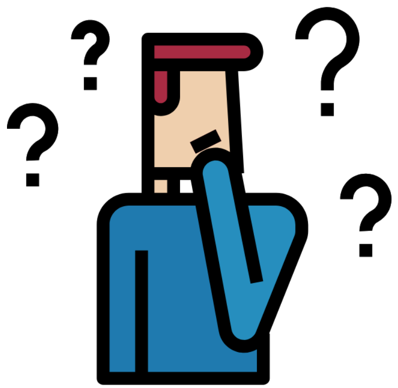
Guarantee the compliance of crowdworking platforms with regulations

- Local (per platform) regulations exist: maximum driving time per day

Transparent and Privacy-preserving regulation enforcement

Collaboration among independent competing platforms

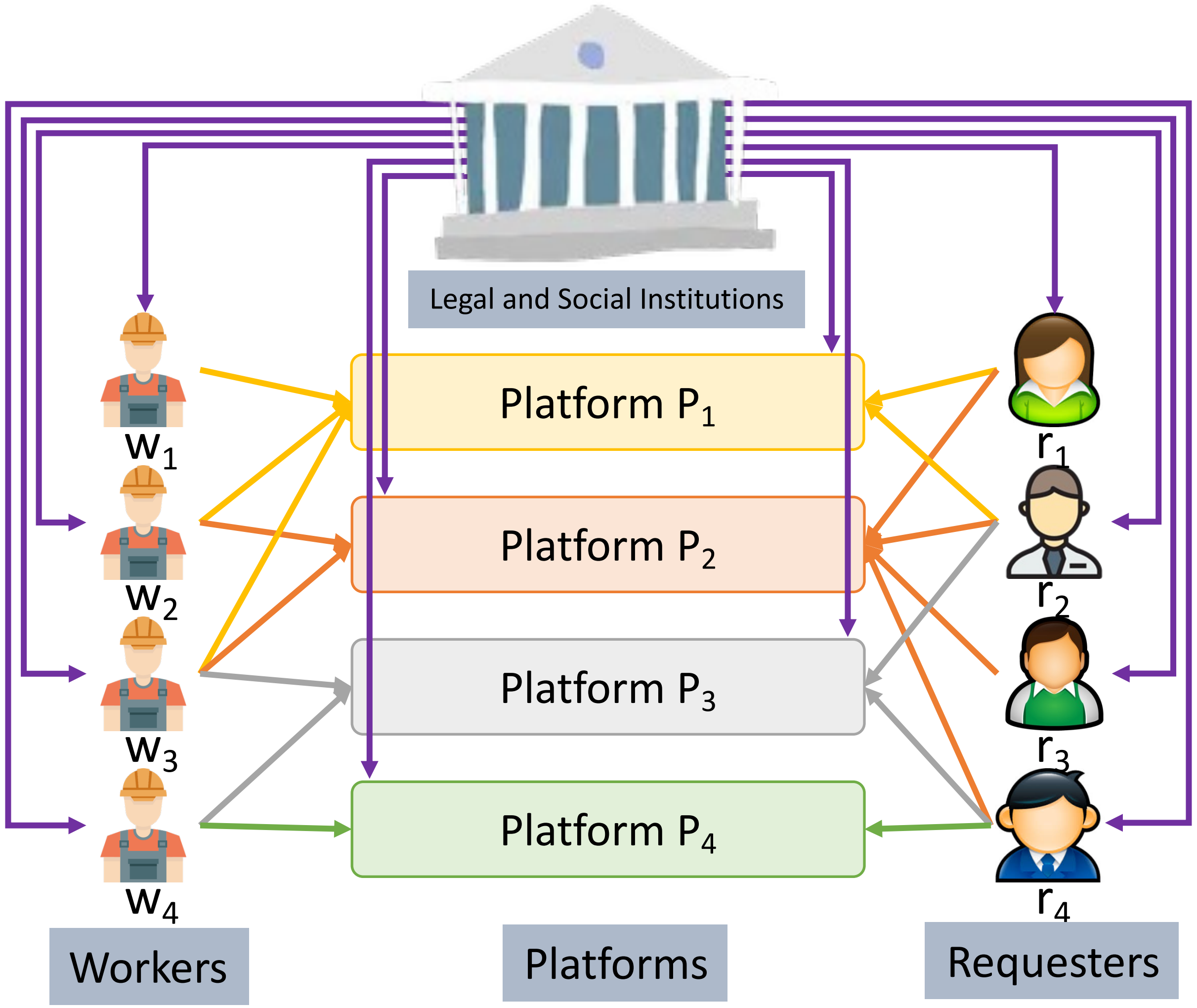
- Enforcement of global regulations
- Complex tasks that may need multiple contributions



Our Vision for Future Regulation Systems

- **Goal: Enforce regulations on multi-platform crowdworking environments while preserving privacy**

- Three main design dimensions
  - D1: Type of supported regulations
    - e.g., aggregate or not
  - D2: Privacy guarantees given to participants
  - D3: Architecture of the system
    - e.g., fully decentralized, partially centralized



Design Space: (D1) Regulation Type

- Express regulations as `SQL` constraints over a universal table `U-TABLE`
- Categorize them according to their `SQL` expression.
- Characterized by:
  - Complexity: `simple` if no `JOIN` operation, `complex` otherwise.
  - Aggregate (presence of Aggregate function(s), with `GROUP BY` and `HAVING`):
  - `row-only`, `aggregate-only` and `mixed`.
- Enforceable: must always hold
  - e.g., maximum work hours
- Verifiable: must hold periodically
  - e.g., minimum work hours

WORKER	PLATFORM	REQUESTER	TIMECOST	CONTRIB
w1	p2	r1	3H	...
w1	p3	r2	2H	...
w2	p1	r1	6H	...

Regulation Example

- r1: the wage proposed by each task must be at least a given amount θ
- A `simple`, `row-only` type of regulation.

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

Regulation Example

- r2: each worker works at most a given amount of time units θ per time period ρ
- A `simple`, `mixed` with `SUM-aggregate` regulation.

```
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(TIMEOUT)  ≥ θ
) );
```

Design Space: (D2) Privacy Guarantees

- Threat model:
  - e.g., honest-but-curious, covert, malicious
  - System-dependent (we do not specify it further)

- Privacy model: pluggable disclosures (to be personalized). We consider:
  - Disclosures to the participants that are not involved in the the crowdworking process π 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 π: $\delta^{\pi}_{R \neg I}$
  - Disclosures to the participants that are directly involved in π (and have thus received task t): δ^{π}_{RI}

Design Space: (D3) Architectural Choices

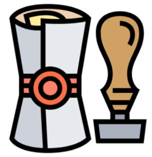
- Any regulation system is made of two critical components:
  - Regulation management: models and enforces the regulations
  - Global state management: stores the global state of the system
- Each component can be implemented either
  - Centralized
    - easier to rapid prototype
    - difficult to ensure fault-tolerance, privacy, and trustworthiness
  - Decentralized
    - more compatible with the multi-platform settings
    - resulting in more overhead and complex communication protocols among entities

SEPAR: a Point in the Design Space

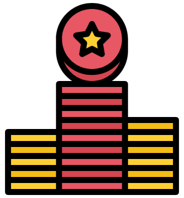
- Regulation supported:
  - `U-TABLE` focuses on the interactions and consists in: `WORKER`, `PLATFORM`, `REQUESTER`, `TIMECOST`.
  - (`simple`, `mixed` with `SUM-aggregate`) regulation, with lower-than (enforceable) or higher-than (verifiable) thresholds.
- Privacy guarantees:
  - Covert non-colluding adversaries.
    - Aims at inferring anything that can be inferred from the execution sequence
    - Is able to deviate from the protocol if no other participant detects it
  - Disclosures sets: (given crowdworking process π: (BEGIN, END, w, p, r, t))
    - $\delta^{\pi}_{\neg R \neg I}$ = (BEGIN, END, p)
    - $\delta^{\pi}_{R \neg I}$ = (BEGIN, END, p, r, t)
    - δ^{π}_{RI} = (BEGIN, END, w, p, r, t)
- Hybrid architecture:
  - Registration Authority (RA): Centralized. Registers participants, models regulations, distributes crypto material.
  - Multi-Platform Infrastructure: Decentralized. Maintains the global state within a blockchain
  - Consensus protocols:
    - Local (nodes of the same platform)
    - Cross-platform (platforms having received the same task)
    - Global (all platforms)

A Simple Token-Based System

- Inspired by e-cash systems, SEPAR implements enforceable and verifiable regulations by managing two budgets per participant
- Lightweight, single-use, and anonymous tokens

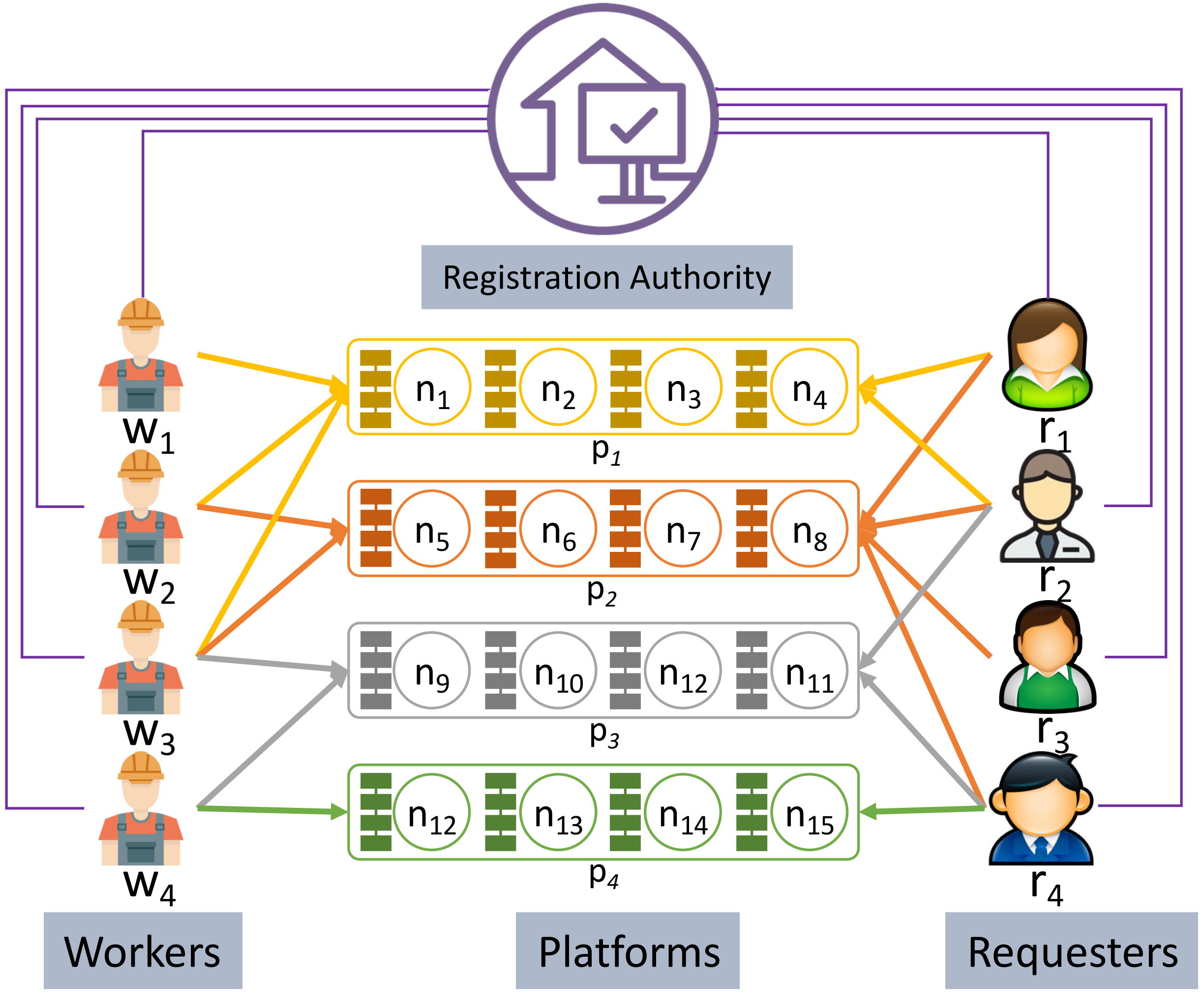


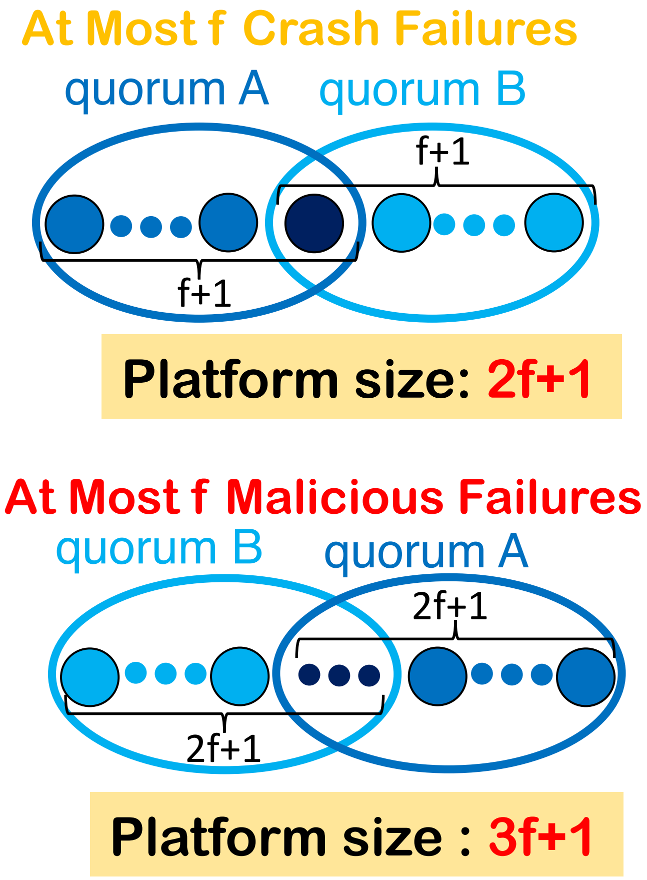
The registration authority refreshes participants tokens periodically



- GENERATE: initializing the budgets and refilling them
  - Enforceable and Verifiable tokens
- SPEND: spending portions of the budgets
- PROVE: providing proof for verifiable regulations to a third party
- CHECK: checking whether a given spending is allowed or not
- ALERT: reporting dubious spending

SEPAR Architecture

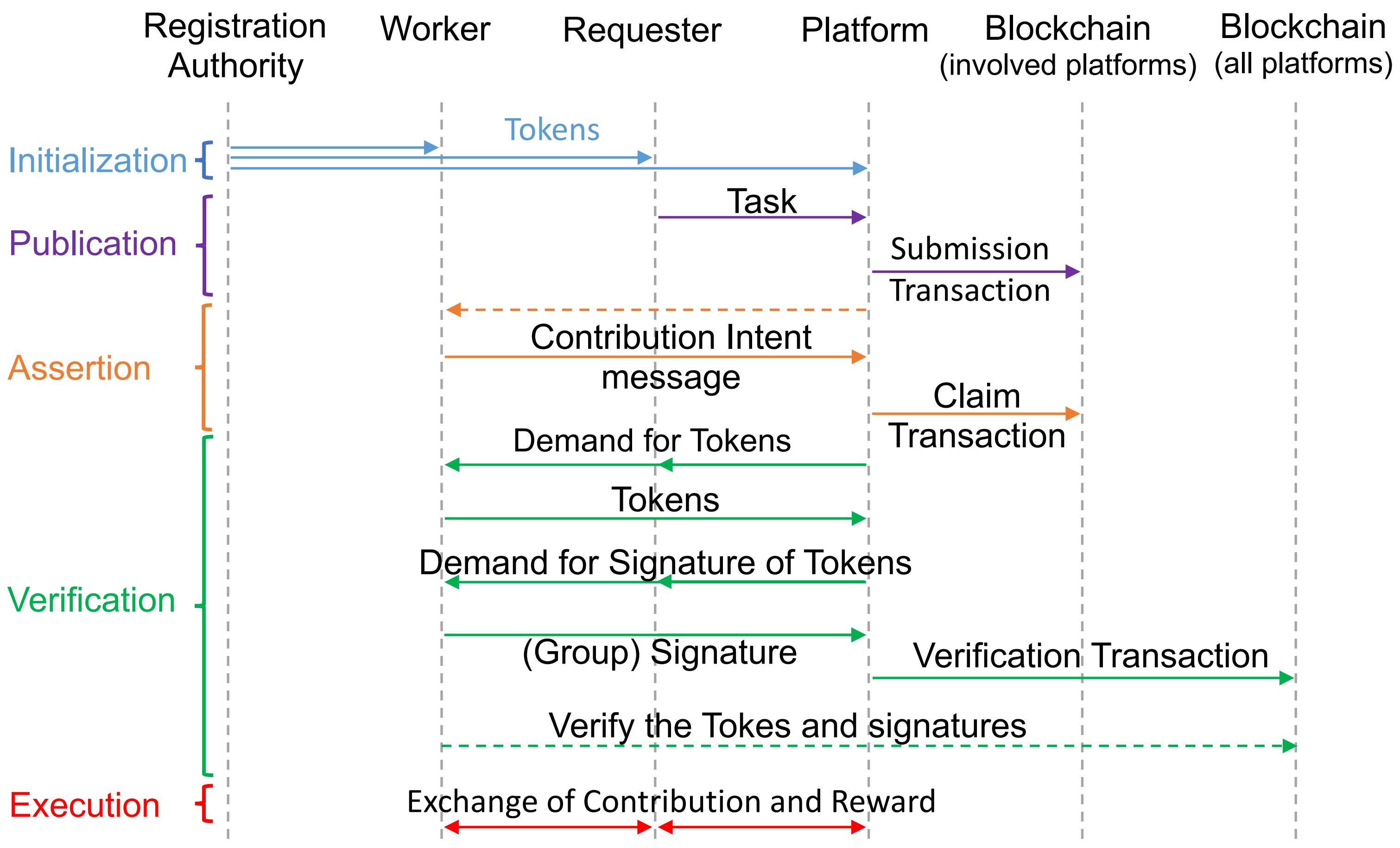




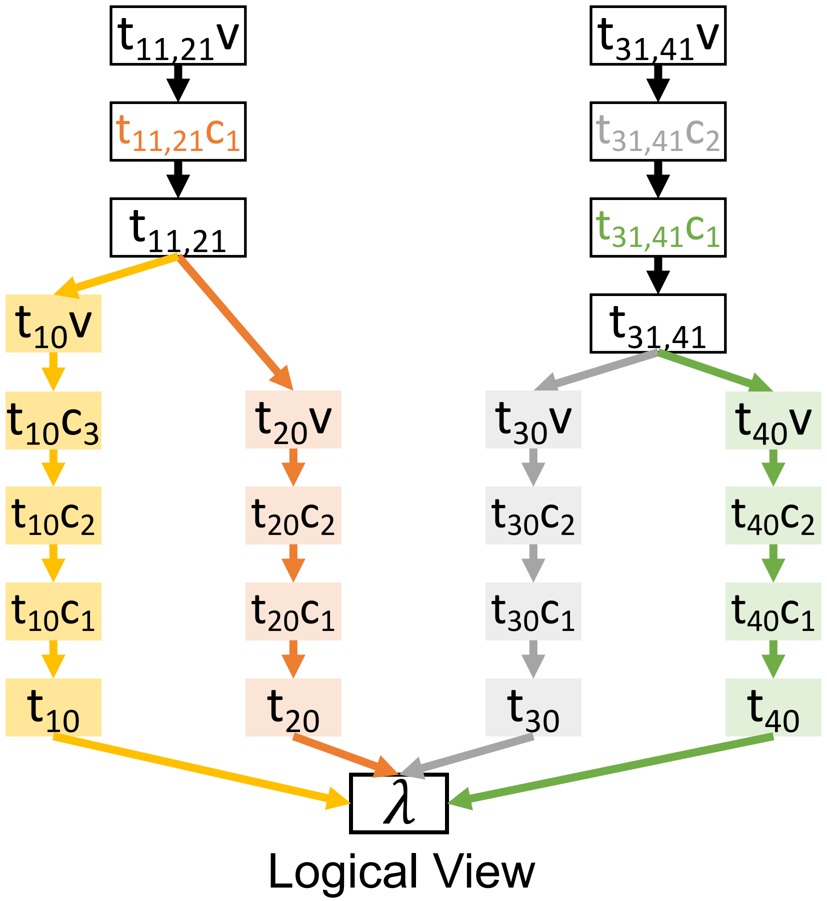
Execution Sequence

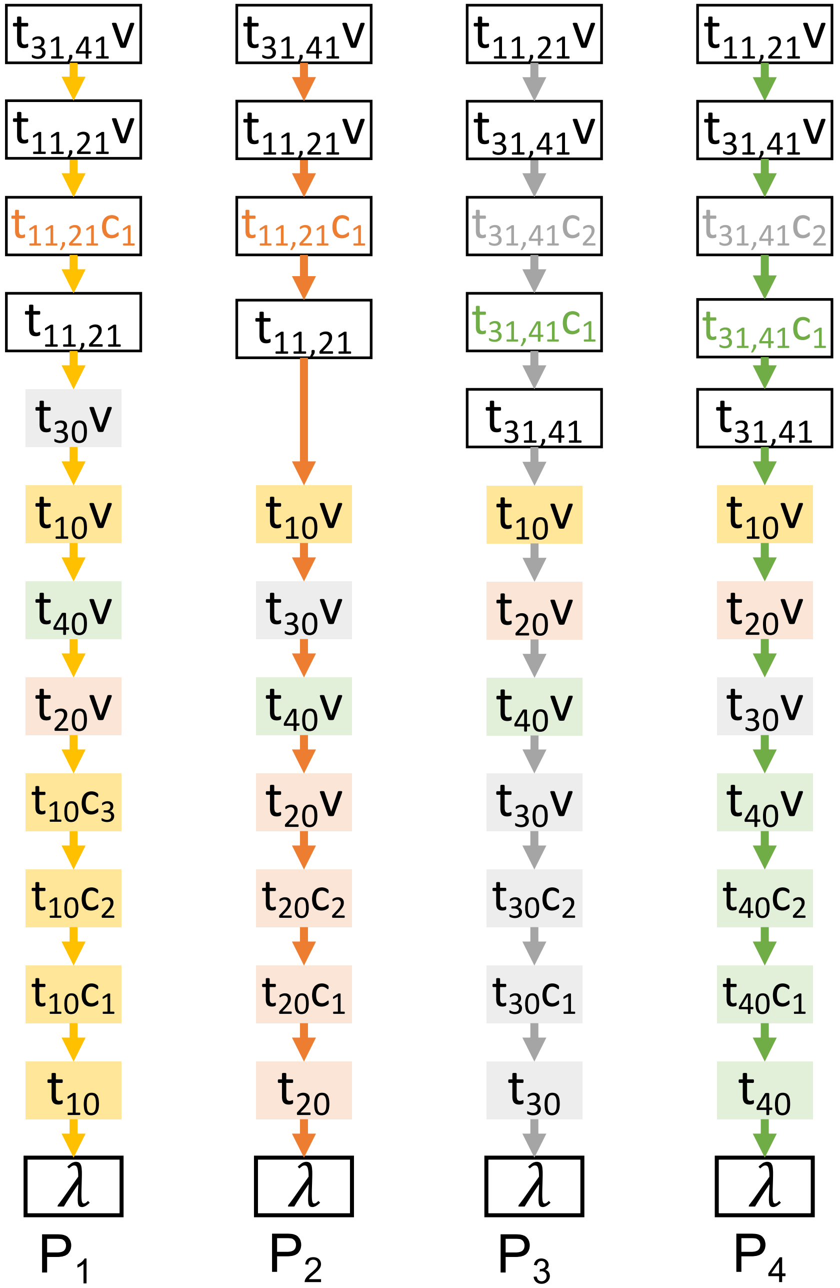
Tasks:
Internal
Cross-Platform

Transactions:
Submission
Claim
Verification

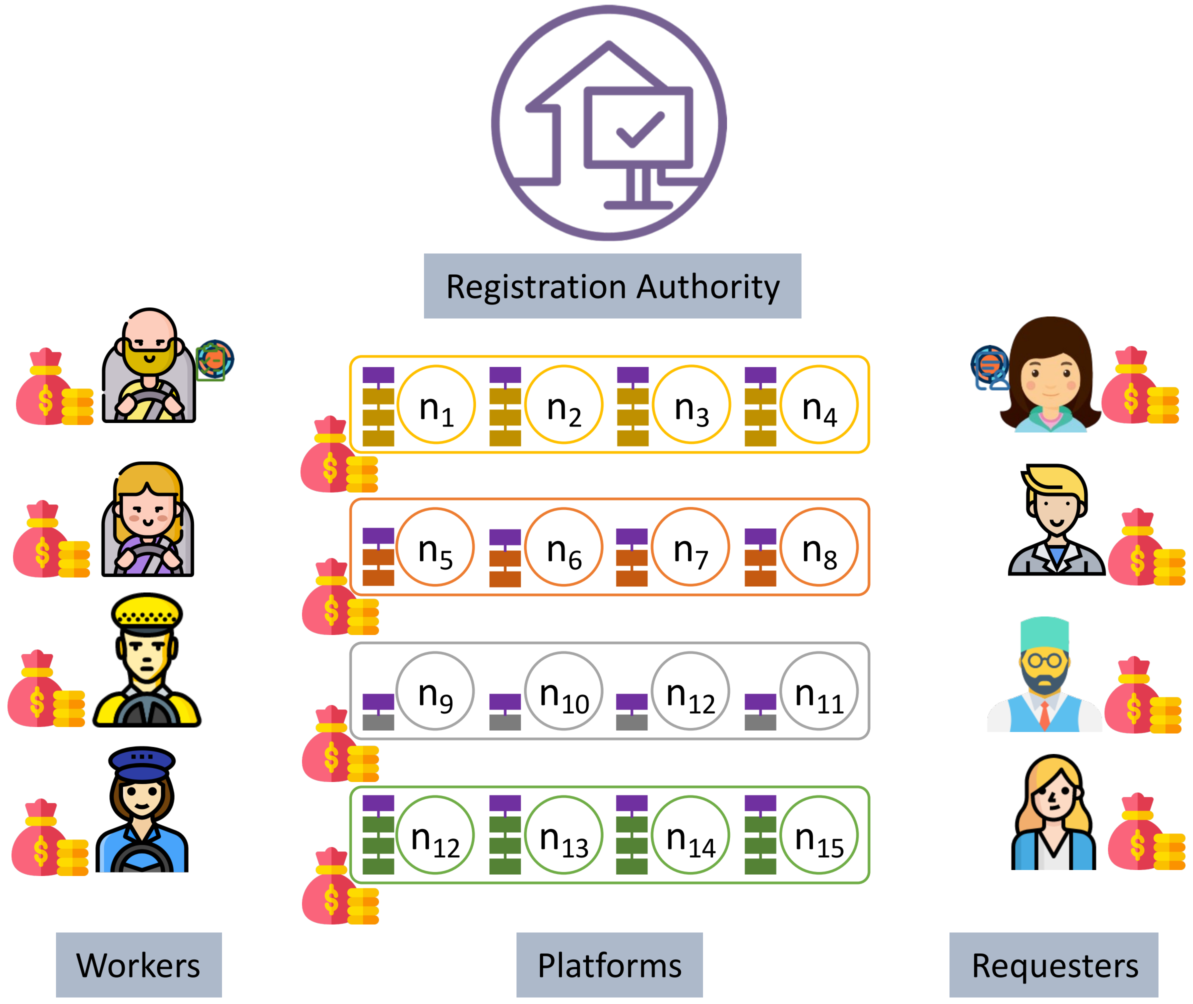


Blockchain Ledger





Processing Tasks in Separ



Consensus in SEPAR

Local Consensus: pluggable and depends on the failure model of nodes

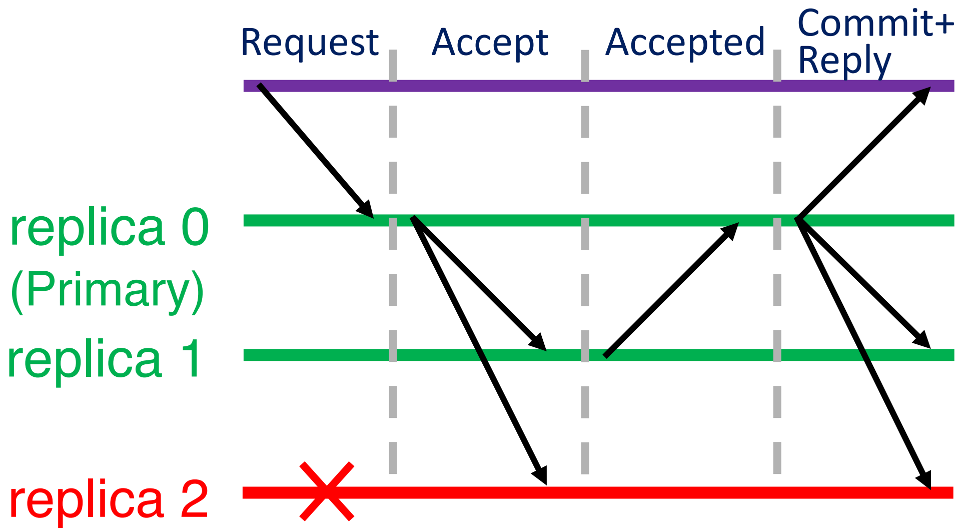
Cross-Platform Consensus: Among the involved platforms

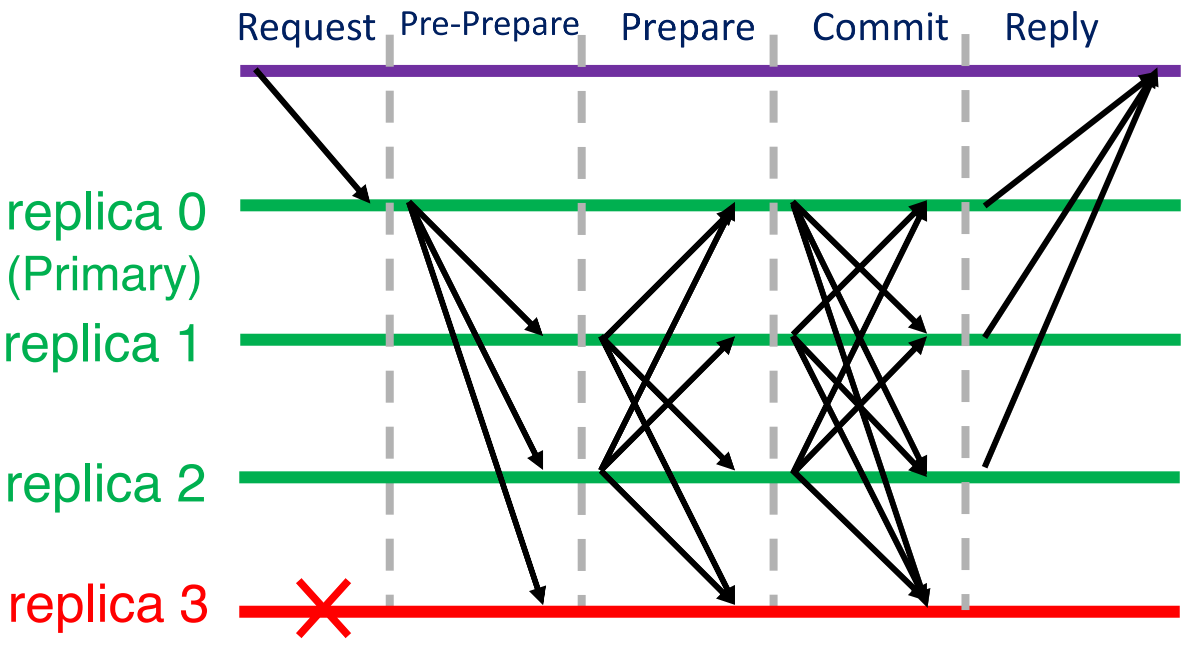
Global Consensus: Requires the participation of all platforms

Transaction/Task	Internal	Cross-Platform
Submission	Local	Cross-Platform
Claim	Local	Cross-Platform
Verification	Global	Global

Local Consensus

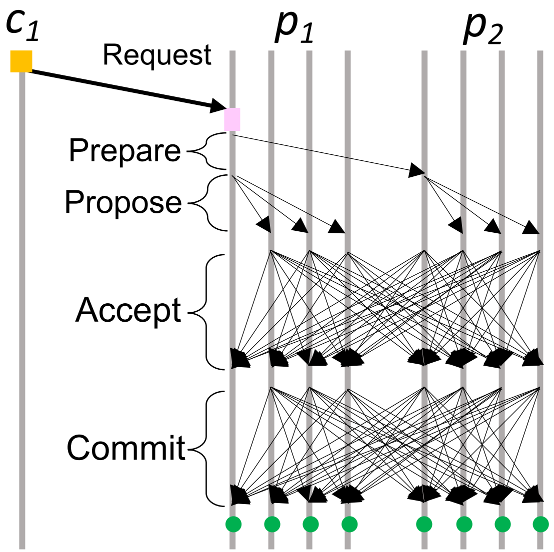
- Required for submission and claim transactions of internal tasks
- Depending on the failure model of nodes
  - Crash failure: (Multi-)Paxos
  - Byzantine failure: PBFT





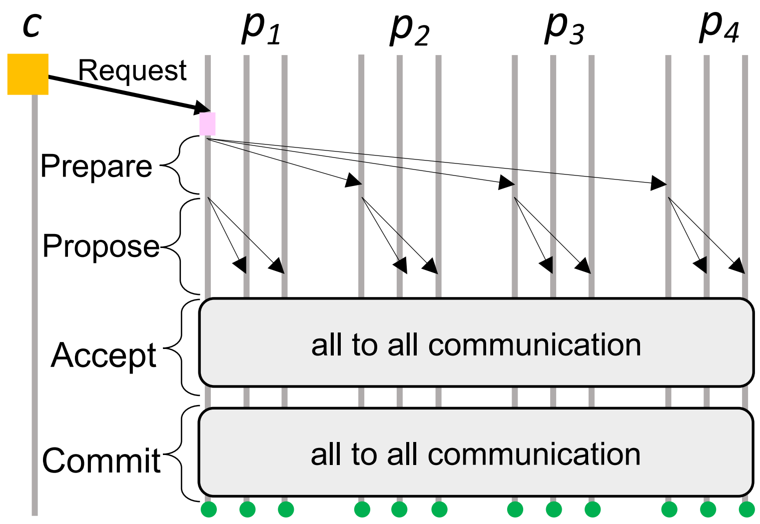
Cross-Platform Consensus

- Is required for submission and claim transactions of cross-platform tasks
- a Byzantine fault-tolerant protocol is used (untrustworthiness of platforms)
- local-majority: required number of matching replies from nodes of a platform
- Each phase: Agreement from the local-majority of all involved platforms



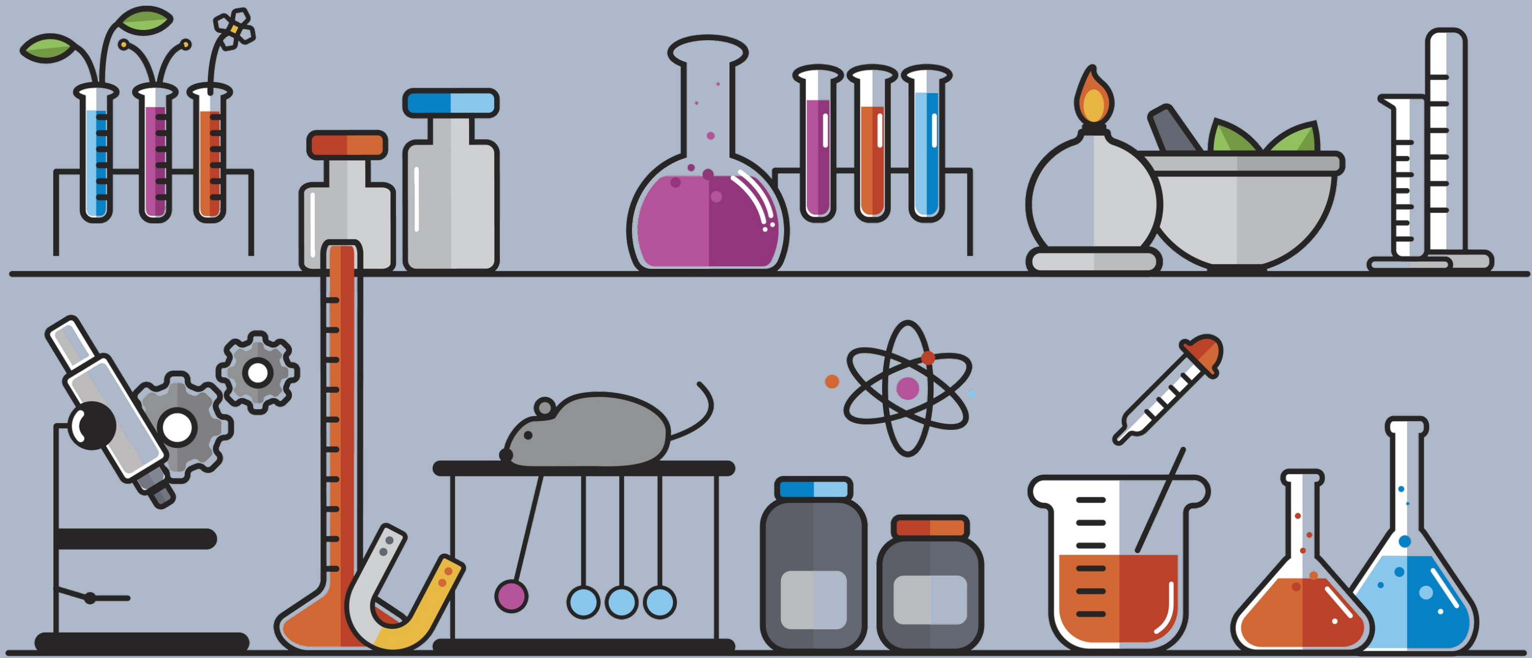
Global Consensus

- Is needed for verification transactions of all tasks
  - Verification transaction: group signatures and all tokens that are consumed by participants to perform a particular task
- A Byzantine fault-tolerant protocol is run among all nodes of every platform
- Each phase: Agreement from the local-majority of two-thirds of the platforms



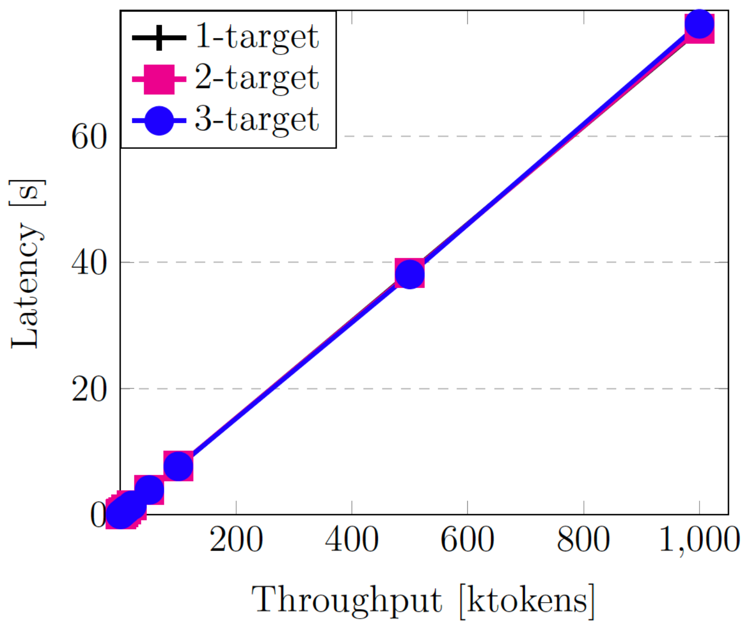
Experimental Settings

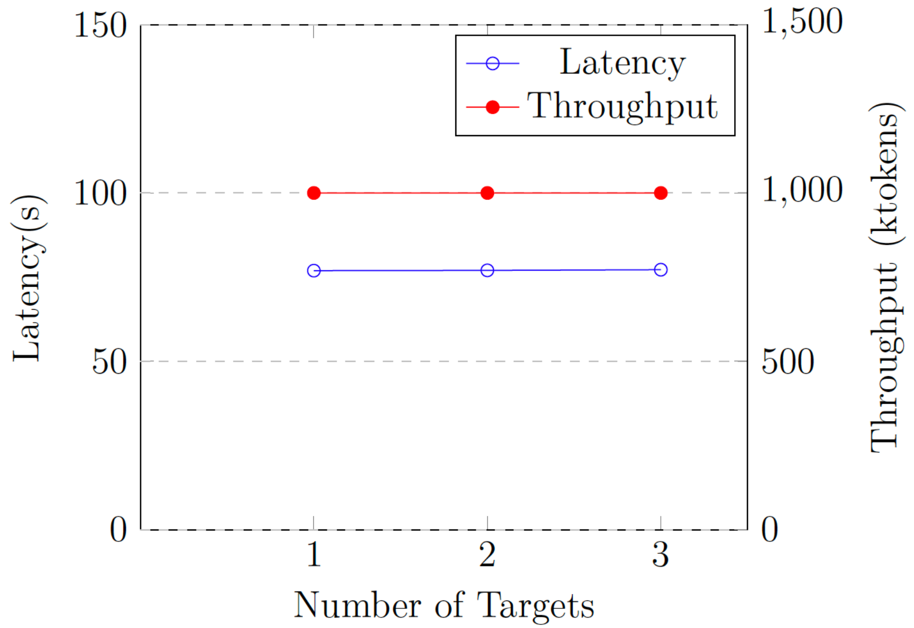
- We do not focus on the description of tasks and contributions
- Platform: Amazon EC2
- Measuring performance
  - Throughput
  - Latency



Token Generation Performance

((w, *, *), θ)
((w, *, r), θ)
((w, p, r), θ)





SEPAR is able to generate tokens in linear time.

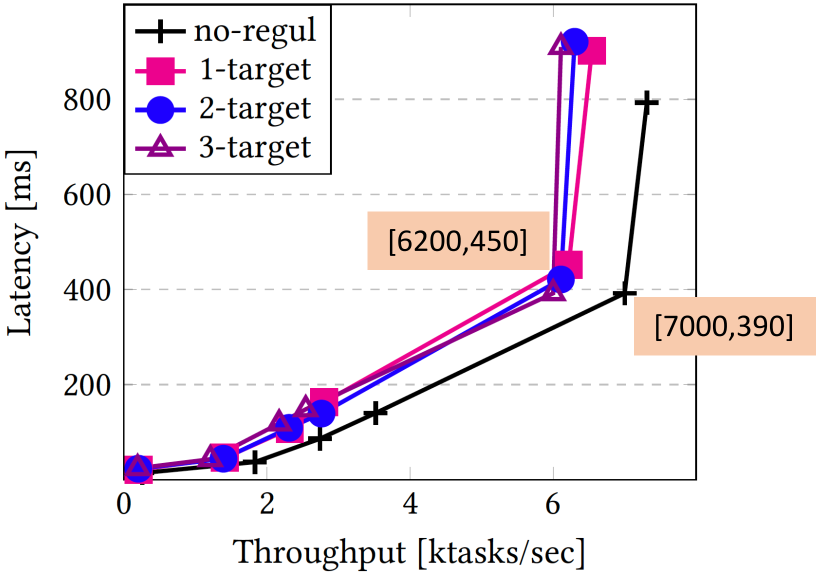
SEPAR generates each token in 0.07ms (1 million tokens in 76 seconds).

The class of regulations does not affect the performance.

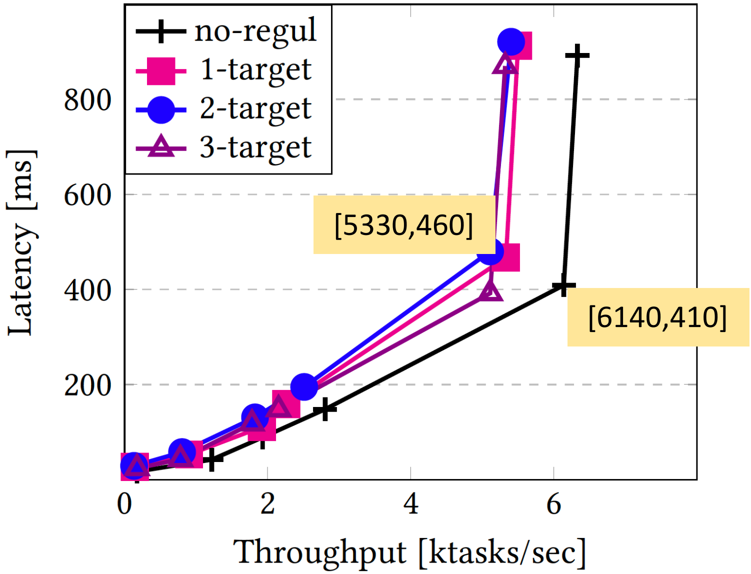
Different Types of Constraints

((w, *, *), θ)
((w, *, r), θ)
((w, p, r), θ)

Crash-Only Nodes



Byzantine Nodes



Four platforms

Two Constraints

10% Cross-Platform

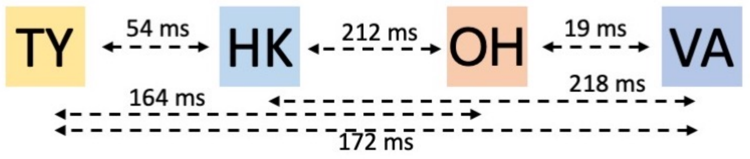
Single Contribution

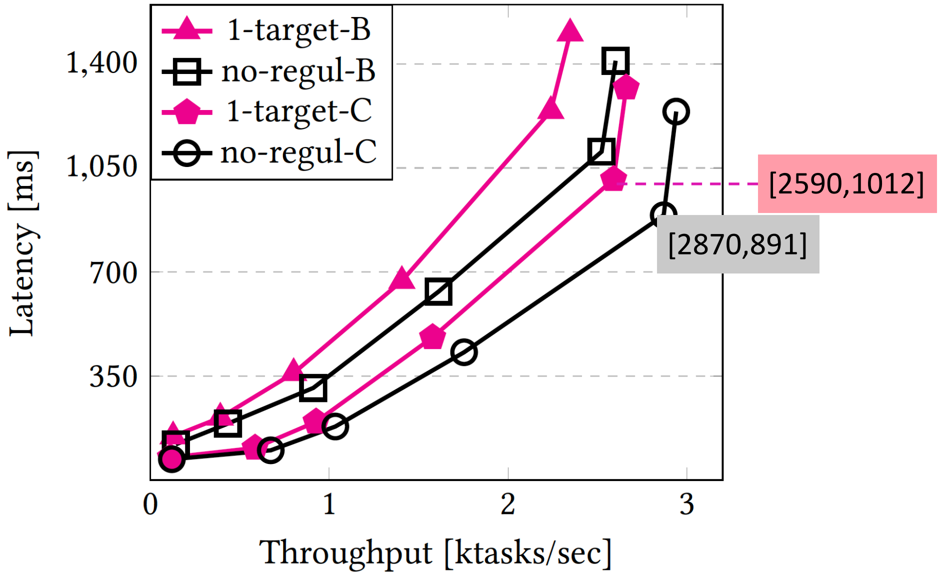
Privacy-preserving mechanism: only 11% throughput and 15% latency overhead (Crash-only)

The class of regulations does not significantly affect the performance of Separ

Scalability Over Spatial Domains

Platforms in different AWS regions:
Tokyo, Hong Kong, Virginia, and Ohio





Four platforms

Two Constraints

10% Cross-Platform

Single Contribution

((w, *, *), θ)

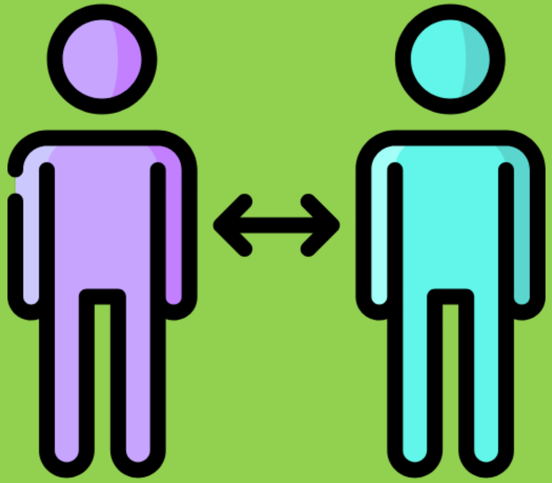
Privacy-preserving mechanism: only 10% throughput and 13% latency overhead (Crash-only)

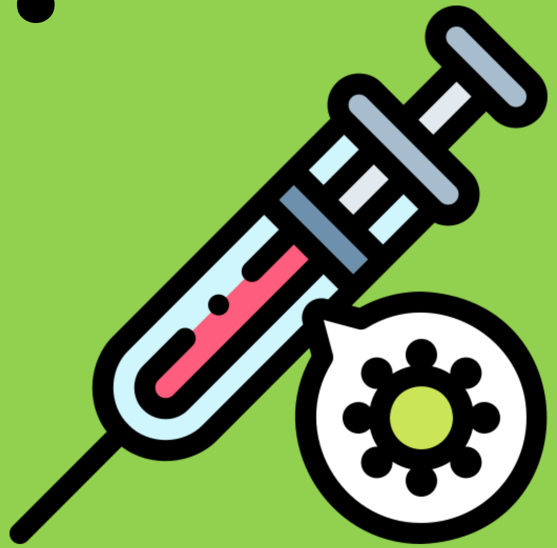
privacy-preserving mechanisms have lower overhead in comparison to a setting with a single data center

SEPAR Conclusion

- An overall vision for future of work multi-platform regulation systems based on three dimensions: Type of Regulations, Privacy, Architecture.
- SEPAR is the first to address the problem of enforcing global regulation over multi-crowdworking platforms.
- A token-based system that enables official institutions to express legal regulations in simple and unambiguous terms
- Supports greater-than and lower-than (`simple`, `SUM-aggregate`) regulations.
- Implemented over a permissioned blockchain that provides transparency using distributed ledgers shared across platforms

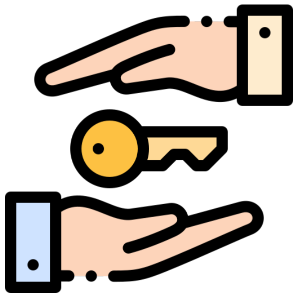
Questions?





Required Cryptographic Material

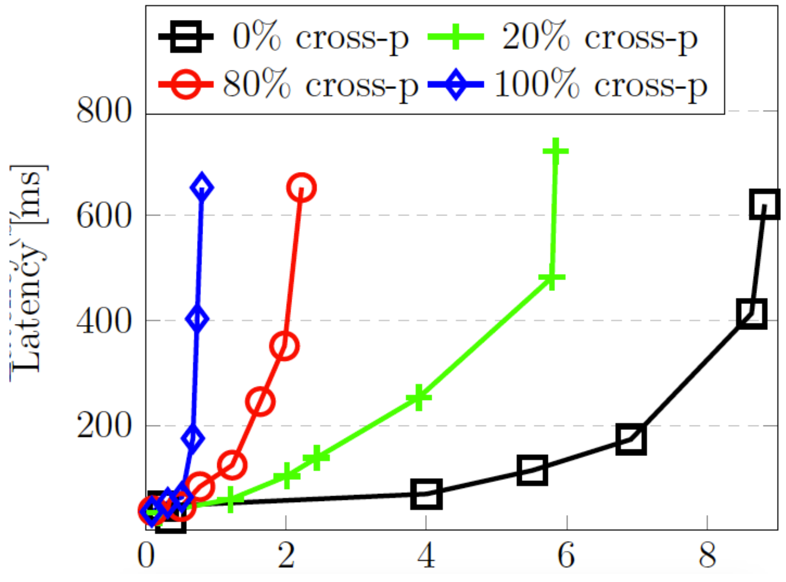
- A pair of usual public/private asymmetric keys (e.g., RSA)
- A pair of public/private asymmetric group keys
  - Union of all workers (Requesters/platforms) forms a group (Group Signature)
  - The registration authority is the group manager
- Participants acquire them when joining SEPAR



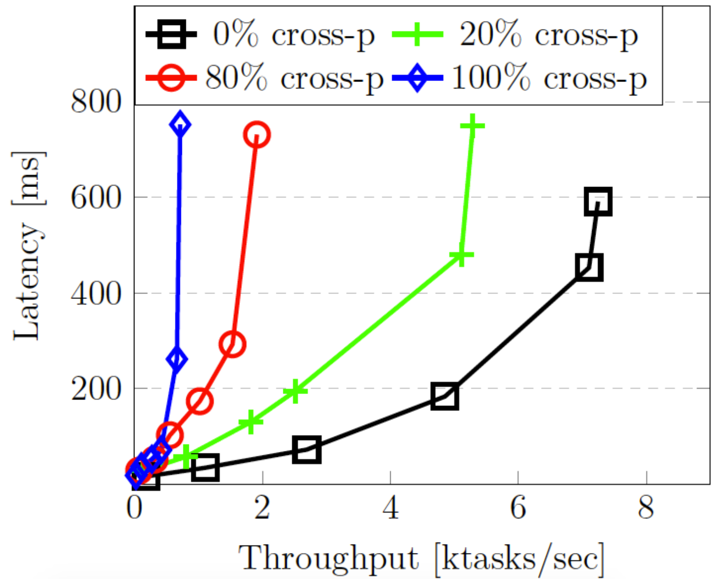
- Signatures:
  1. the group signature of the token(which will be later verified by all platforms, together with the token, when it is shared with all platforms),
  2. the group signature of the pair consisting of a token and a task.

Increasing the Number of Cross-Platform Tasks

Crash-Only Nodes



Byzantine Nodes



Four platforms

Two Constraints

Single Contribution

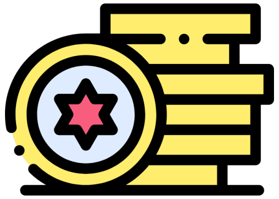
SEPAR is able to process 8600 local tasks with 400 ms latency

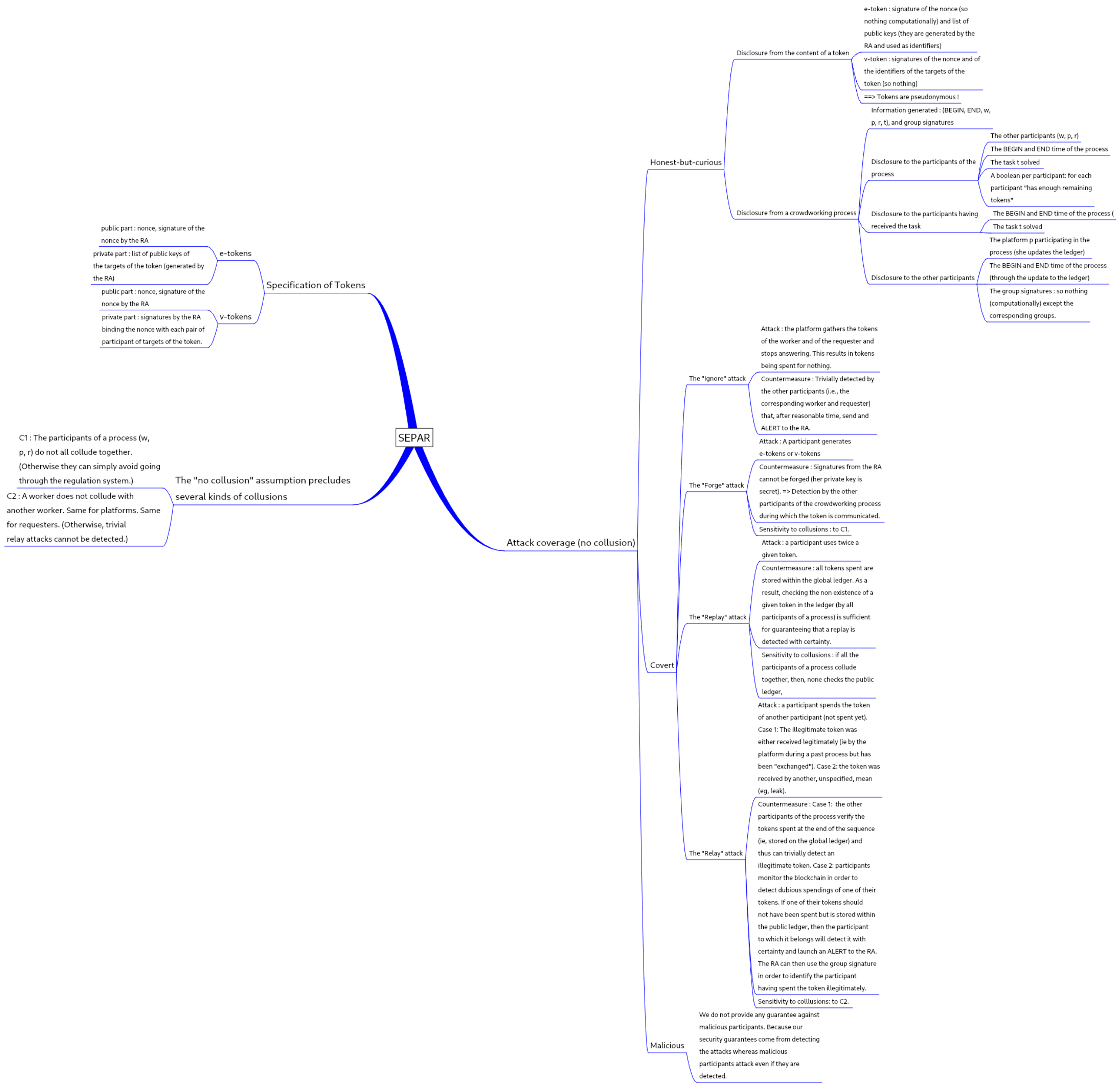
Increasing the percentage of cross-platform tasks, reduces the overall throughput

Constraint and certificate tokens

- Enforceable Token:
  - is a tuple (t_p, t_s) where
  - t_p = (nonce, (nonce)$\sigma_{PK(RA)}$)
  - t_s is the list of public keys of involved participants

- Verifiable Token:
  - is a tuple (t_p, t_s) where
  - t_p = (nonce, (nonce)$\sigma_{PK(RA)}$)
  - t_s is (nonce, o,w) $\sigma_{PK(RA)}$, (nonce, o,p) $\sigma_{PK(RA)}$, and (nonce, o,r) $\sigma_{PK(RA)}$
  - Where o is the identity of the participant owner of the token




SEPAR
Specification of Tokens
e-tokens
public part : nonce, signature of the nonce by the RA
private part : list of public keys of the targets of the token (generated by the RA)
v-tokens
public part : nonce, signature of the nonce by the RA
private part : signatures by the RA binding the nonce with each pair of participant of targets of the token.
The "no collusion" assumption precludes several kinds of collusions
C1 : The participants of a process (w, p, r) do not all collude together. (Otherwise they can simply avoid going through the regulation system.)
C2 : A worker does not collude with another worker. Same for platforms. Same for requesters. (Otherwise, trivial relay attacks cannot be detected.)
Attack coverage (no collusion)
Honest-but-curious
Disclosure from the content of a token
e-token : signature of the nonce (so nothing computationally) and list of public keys (they are generated by the RA and used as identifiers)
v-token : signatures of the nonce and of the identifiers of the targets of the token (so nothing)
==> Tokens are pseudonymous !
Disclosure from a crowdworking process
Information generated : (BEGIN, END, w, p, r, t), and group signatures
Disclosure to the participants of the process
The other participants (w, p, r)
The BEGIN and END time of the process
The task t solved
A boolean per participant: for each participant "has enough remaining tokens"
Disclosure to the participants having received the task
The BEGIN and END time of the process (
The task t solved
Disclosure to the other participants
The platform p participating in the process (she updates the ledger)
The BEGIN and END time of the process (through the update to the ledger)
The group signatures : so nothing (computationally) except the corresponding groups.
Covert
The "Ignore" attack
Attack : the platform gathers the tokens of the worker and of the requester and stops answering. This results in tokens being spent for nothing.
Countermeasure : Trivially detected by the other participants (i.e., the corresponding worker and requester) that, after reasonable time, send and ALERT to the RA.
The "Forge" attack
Attack : A participant generates e-tokens or v-tokens
Countermeasure : Signatures from the RA cannot be forged (her private key is secret). => Detection by the other participants of the crowdworking process during which the token is communicated.
Sensitivity to collusions : to C1.
The "Replay" attack
Attack : a participant uses twice a given token.
Countermeasure : all tokens spent are stored within the global ledger. As a result, checking the non existence of a given token in the ledger (by all participants of a process) is sufficient for guaranteeing that a replay is detected with certainty.
Sensitivity to collusions : if all the participants of a process collude together, then, none checks the public ledger,
The "Relay" attack
Attack : a participant spends the token of another participant (not spent yet). Case 1: The illegitimate token was either received legitimately (ie by the platform during a past process but has been "exchanged"). Case 2: the token was received by another, unspecified, mean (eg, leak).
Countermeasure : Case 1: the other participants of the process verify the tokens spent at the end of the sequence (ie, stored on the global ledger) and thus can trivially detect an illegitimate token. Case 2: participants monitor the blockchain in order to detect dubious spendings of one of their tokens. If one of their tokens should not have been spent but is stored within the public ledger, then the participant to which it belongs will detect it with certainty and launch an ALERT to the RA. The RA can then use the group signature in order to identify the participant having spent the token illegitimately.
Sensitivity to colllusions: to C2.
Malicious
We do not provide any guarantee against malicious participants. Because our security guarantees come from detecting the attacks whereas malicious participants attack even if they are detected.