

# CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing

Hailiang ZHAO  
<http://hliangzhao.me>

December 10, 2019

This paper was published on **IEEE Trans. on Parallel and Distributed Systems (TPDS)**, Jun 2019.

# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Outline

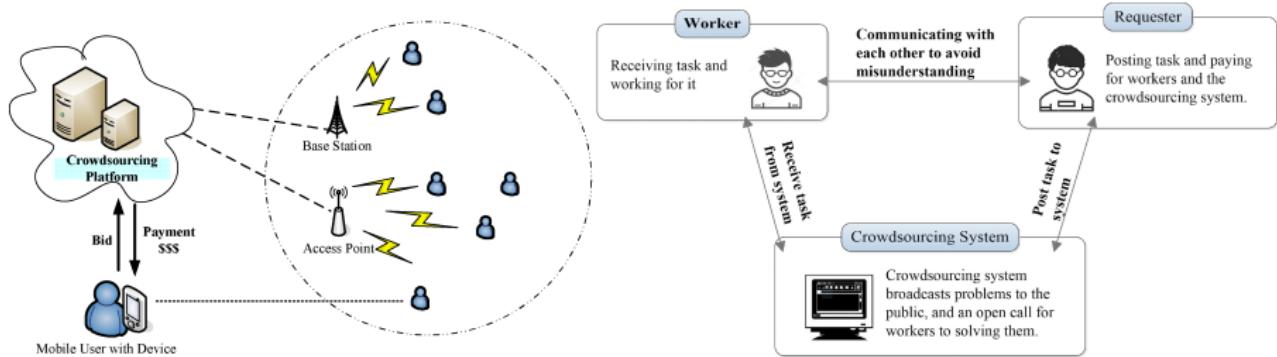
- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# What is Crowdsourcing?

## Crowdsourcing

It is a distributed problem-solving model through an open call for solutions, which consists of three groups of roles:

- ① **requesters**
- ② **workers**
- ③ **a centralized crowdsourcing system**



# The Weakness of Traditional TRUST-based Model

## ① Vulnerable to DDoS attacks

Elance and oDesk downed services for many workers due to be hit by DDoS attacks in May 2014.

## ② Single point of failure

Uber China's passengers can't stop the order because of hardware failure, April 2015.

## ③ Risk of privacy disclosure and data loss

Freelancer breached the PRIVACY ACT for uncovering a user's true identity, Dec 2015.

## ④ False-reporting

A subjective arbitration could happen.

## ⑤ Royalties from platforms

Most of crowdsourcing systems demands a sliding services fee for 5% ~ 20%.

## How to Deal with the above Issues?

- ① **Data security**: Encryption and differential privacy
- ② **False-reporting**: Reputation-based mechanisms
- ③ **Single point of failure & performance bottleneck**:  
Distributed architectures

Those approaches cannot solve the issues **simultaneously** and suffer from **breakdown of trust**.

Can We Design A Decentralized Crowdsourcing System with Reliability, Fairness, Security, and Low Service Fee?



**Let us embrace the future of Blockchain**

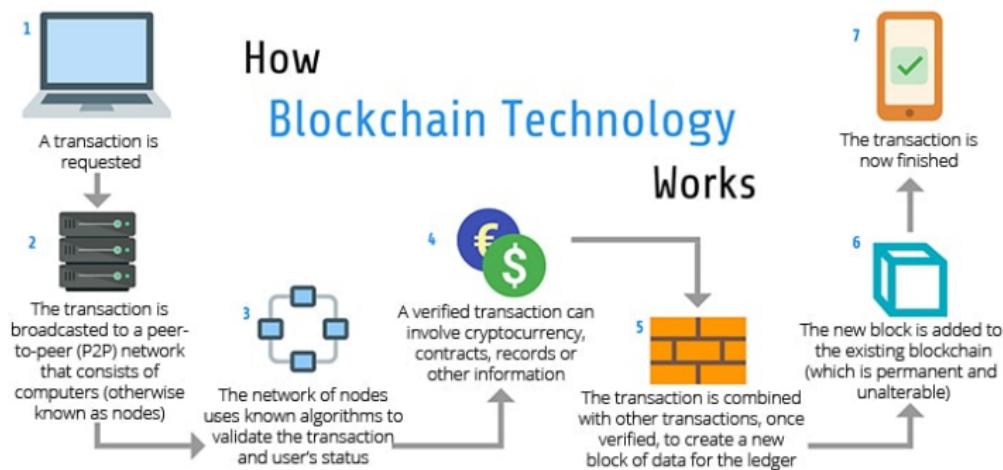
# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# What is Blockchain?

**Blockchain** reduces the existence of third-party intermediary

It is a **decentralized** architecture and **distributed** computing paradigm underlying Bitcoin, whose key advantages include **decentralization, time-series data, collective maintenance, programmability, and security**.

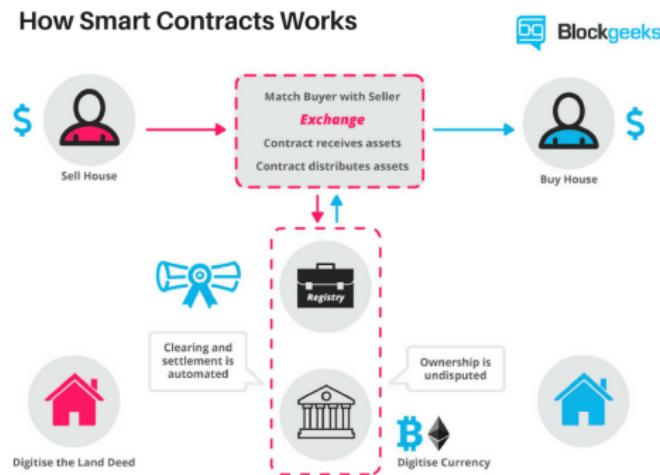


# What is Smart Contract?

## Smart Contract

They are computerized transaction protocols that can **self-verify** and **self-execute** the terms of contracts **without** a third-party intermediary.

**Decentralized + Trustless + Programmable + Tamper-resistant**



# Outline

1

## Introduction

2

## Background

- Blockchain
- Smart Contract

3

## Theoretical Model

- Roles
- Potential Threats and Malicious Behaviors
- Security Assumption

4

## Framework Design

- Three Layers Architecture
- Crowdsourcing Procedure in CrowdBC

5

## Concrete Implementation

- Designed Smart Contracts
- Time-locked Deposit Protocol
- Reputation Management

# Four Roles: Requester, Worker, CrowdBC Client, Miner

## ① Requesters

- ① Identified by  $\mathcal{R} \triangleq \{R_1, \dots, R_n\}$ , post tasks by providing descriptions
- ② Deposite (reward + monetary penalty) cannot be redeemed before ddl

## ② Workers

- ① Identified by  $\mathcal{W} \triangleq \{W_1, \dots, W_m\}$ , compete for tasks to get rewards
- ② Worker  $W_j$  is selected depending on a tuple of reliability value

$$\{\beta_{W_j}, \mathcal{C}_k(\epsilon_k, \psi_k)\}$$

- ③ With reputation  $\beta_{W_j}$ ,  $W_j$  submitt solutions to  $\mathcal{C}_k$  for  $\epsilon_k$  times and got  $\psi_k$  high evaluation
- ④ A deposit  $\pi_{W_j}$  is required to thwart DDoS and Sybil attack (**How?**)
- ⑤ The solutions submitted by workers are identified by  $s \triangleq \{s_1, \dots, s_m\}$

# Four Roles: Requester, Worker, CrowdBC Client, Miner

## ③ CrowdBC Client

- ① Served as a medium for  $\mathcal{R}$  and  $\mathcal{W}$ , is not controlled by any third party
- ② Run locally on users' personal computer

## ④ Miners

- ① Add past transactions into a block and provide a valid proof (through mining) to claim the block reward and transaction fees
- ②  $\mathcal{R}$  and  $\mathcal{W}$  can also be miners if they participate in the mining

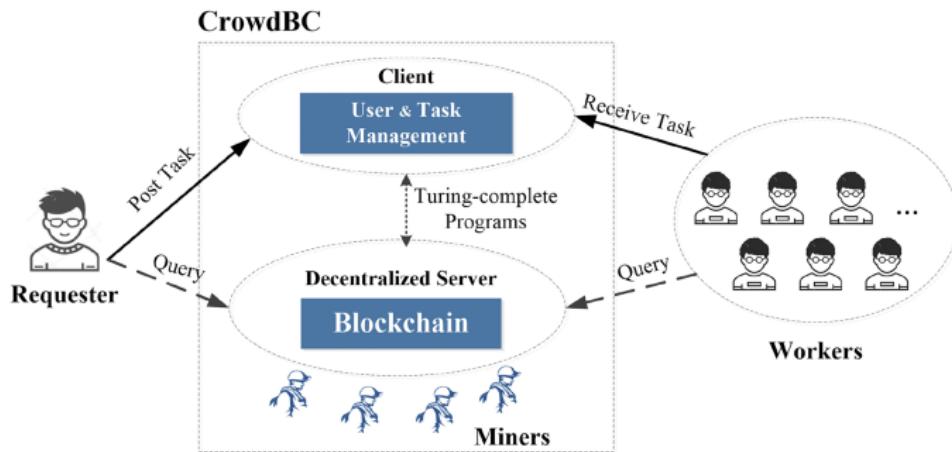
2 conditions for 'winning' a block



# System Model Overview

## Scheme

- ▷  $\mathcal{R}$  and  $\mathcal{W}$  firstly need to register:  $(K_{R_i}^p, K_{R_i}^s, K_{R_i}^a), (K_{R_i}^p, K_{R_i}^s, K_{R_i}^a)$ .
- ▷  $R_i$  posts a task with **CrowdBC Client** (the categories are pre-defined)
- ▷ Authenticated  $W_j$  can receive the task
- ▷  $\mathcal{R}$  reach agreements with  $\mathcal{W}$  by smart contracts in CrowdBC



# Potential Threats

## ① Malicious Requesters

- ① Collect useful solutions without losing deposit [**false-reporting**]
- ② Misreport the high-quality solution as low level
- ③ Deny the obtain of solutions
- ④ Create a fork chain after receiving the solutions

## ② Malicious Workers

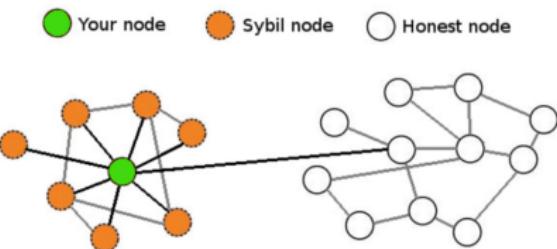
- ① Obtain rewards without paying sufficient effort [**free-riding attack**]
- ② Create fork chain when receiving low level evaluation
- ③ receive tasks but not submit solutions on time

## ③ Malicious Miners

- ① Earn virtual coins by forking a chain
- ② Collude with malicious requesters/workers to break the normal execution

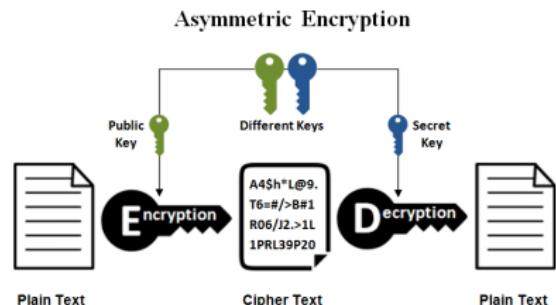
# Security Assumption

## ① Majority Honest Security



## ② Secure Transfer with Wallet

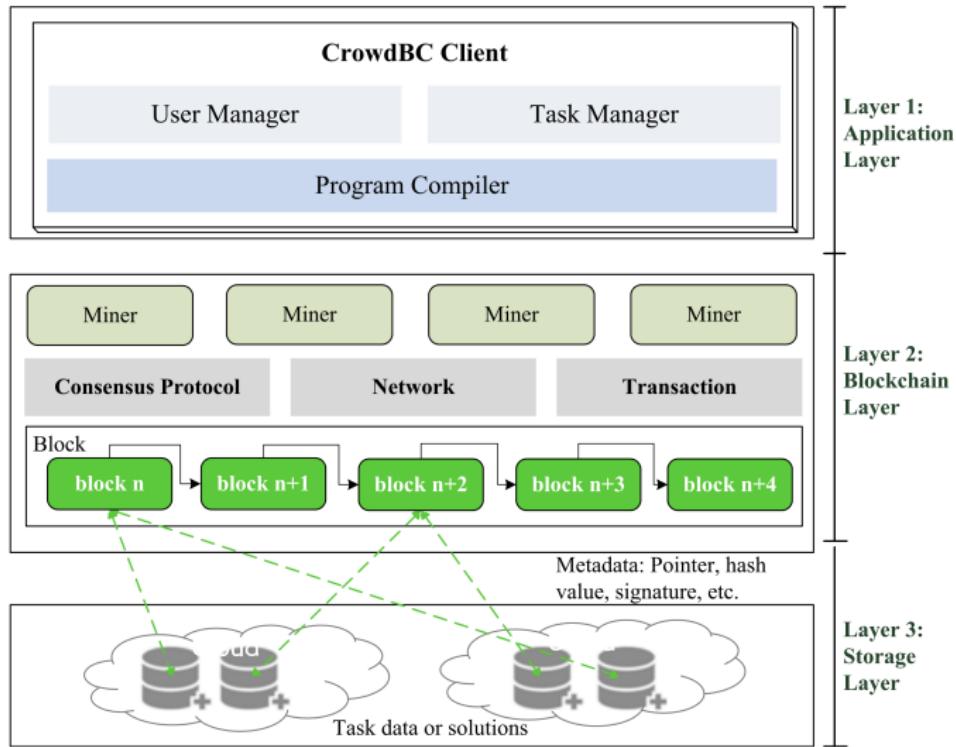
## ③ Secure Encryption Algorithm



# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Three Layers Architecture of CrowdBC



# Three Layers Architecture of CrowdBC

## ① Application Layer [CrowdBC Client]

- ① The entrance for users to finish a crowdsourcing task
- ② **User Manager (UM)**: Registration and user info. management
- ③ True identities are not needed, just public key and address are fine
- ④ Registered users can post or receive tasks with **Task Manager (TM)**
- ⑤ TM manages task posting, task receiving, solution submission, and reward assignment

## ② Blockchain Layer [Decentralized Server]

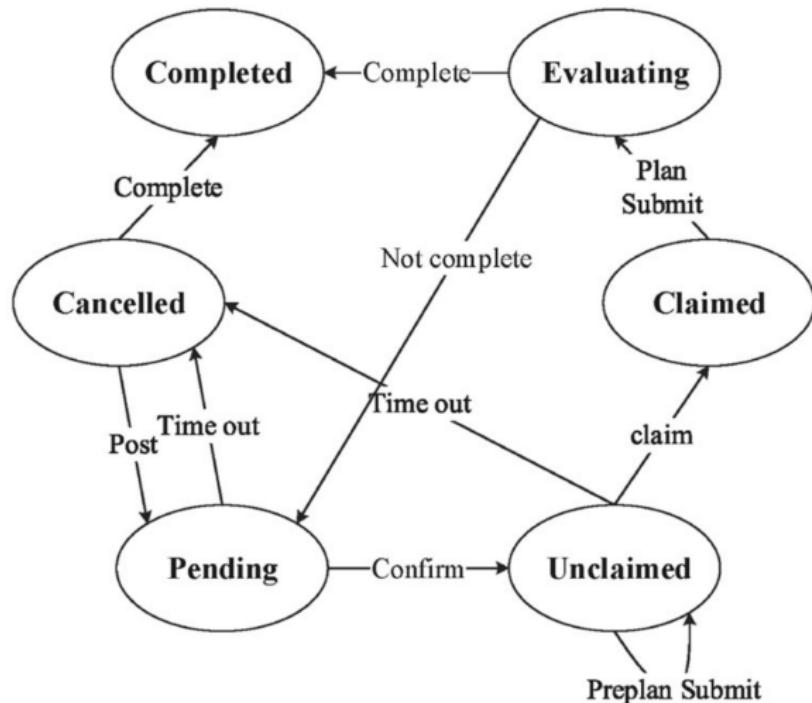
- ① Provide consensus on the order in which program is written
- ② Run state machine for tasks

## ③ Storage Layer [Distributed Database]

- ① Data values are signed by private keys of data owners
- ② Authenticity and integrity of the data values can be checked
- ③ Task data are stored **outside** the blockchain

# State Machine Construction for Tasks

State Machine depicts the life cycle of each task's processing.



# Crowdsourcing Process in CrowdBC

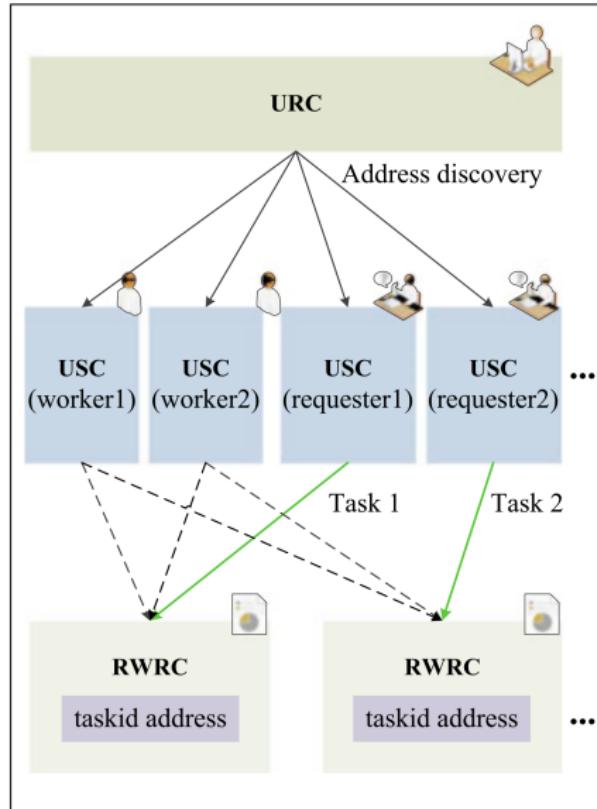
- ① **Requesters and Workers** register in CrowdBC  
(as a **transaction** contained user info., done by **CrowdBC Client**)
- ② **Miners** confirm the **transaction**  
(confirmed data and status are stored into the blockchain permanently)
- ③ **Requesters** post tasks  
(**requesters** also need provide *reward*, *deposit*, and *an evaluation function* in advance)
- ④ **Workers** choose tasks  
(coins or reputation value is deposited)
- ⑤ **Workers** submit solutions before ddl  
(solution is encrypted with **requester**'s public key and sent to the distributed storage)
- ⑥ Task evaluation, reward assignment, and reputation renewal  
(according to the evaluation results)

# Outline

- 1 Introduction
- 2 Background
  - Blockchain
  - Smart Contract
- 3 Theoretical Model
  - Roles
  - Potential Threats and Malicious Behaviors
  - Security Assumption
- 4 Framework Design
  - Three Layers Architecture
  - Crowdsourcing Procedure in CrowdBC
- 5 Concrete Implementation
  - Designed Smart Contracts
  - Time-locked Deposit Protocol
  - Reputation Management

# Designed Smart Contracts: An Overview

| User Register Contract (URC)                  |              |             |
|---|--------------|-------------|
| Alice   | Address      | USC address |
| Bob   | Address      | USC address |
| ...   |              |             |
| User Summary Contract (USC)                   |              |             |
| Alice address                                 |              |             |
| Profile                                       | Reputation   |             |
| Expertise                                     | Activity     |             |
| Task list                                     |              |             |
| id1   | RWRC address | Status      |
| id2   | RWRC address | Status      |
| ...   |              |             |
| Requester-Worker Relationship Contract (RWRC) |              |             |
| Owner   | Status       |             |
| Deposit                                       | Evaluation   |             |
| Taskid address                                |              |             |
| Task pointer                                  |              |             |



# Designed Smart Contracts

## ① User Register Contract (URC)

- ① Requesters & Workers do not need to submit true identities
- ② Requesters & Workers will be assigned with a key pair
- ③ The procedure is processed as a transaction
- ④ Registration needs transaction fee for miners to confirm the transaction

## ② User Summary Contract (USC)

- ① It stores personal statistics information of requesters & workers
- ② Task status includes *Pending*, *Unclaimed*, and *Finished*

## ③ Requester-Worker Relationship Contract (RWRC)

- ① It depicts the agreement between requesters and workers
- ② It is created when a task  $T$  is posted

$$T \triangleq \{desc, K_R^p, coins(v + \pi_R), (\beta_k, \epsilon_k, \psi_k), \lambda, \tau_{ddl}, \tau_{confirm}, eval(\cdot)\}$$

- ③ It checks workers' reputation can be satisfied or not to receive the task
- ④ Qualified workers add their addrs to the corresponding worker pool
- ⑤ How to evaluate the solutions are written in RWRC

# Time-locked Deposit Protocol

Participants are required to make a **time-locked deposit** as a guarantee to regulate their behavior.

## Time-locked Deposit Protocol

**Phase #1: RDeposit.** Upon receiving deposit from  $R_i$ , record the corresponding info. into blockchain by creating **RWRC**. The deposit is locked until  $\tau_{ddl}$  or the task is received by some worker.

**Phase #2: WDeposit.** Before  $\tau_{ddl}$ ,  $W_j$  check if the remaining deposit is not redeemed and receive the task if qualified.

**Phase #3: Claim.** Before  $\tau_{ddl}$ ,  $W_j$  submits a solution to storage layer and give the addr to  $R_i$ . After confirmed by  $R_i$ ,  $W_j$  submits the claim transaction in **RWRC** and his deposit can be redeemed.

**Phase #4: Reward.** Before  $\tau_{confirm}$ ,  $R_i$  start transaction to reward workers according to their evaluation results. After the transaction being confirmed,  $R_i$  redeems his deposit.

# Reputation Management

For each work  $W_j$ , after finishing a task, his reputation value will be updated by

$$\beta_W = \begin{cases} \min(\beta_W^{Max}, \beta_W + 1), & \text{if } a = H \text{ and } \beta_W \geq h_k \\ \beta_W - 1, & \text{if } a = L \text{ and } \beta_W \geq h_k + 1 \\ 0, & \text{if } a = L \text{ and } \beta_W = h_k \\ \beta_W + 1, & \text{if } \beta_W < h_k + 1, \end{cases}$$



# CrowdBC Workflow

