CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing

Hailiang ZHAO
http://hliangzhao.me

Decemeber 10, 2019

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

Outline

Outline

Outline

Outline

Outline

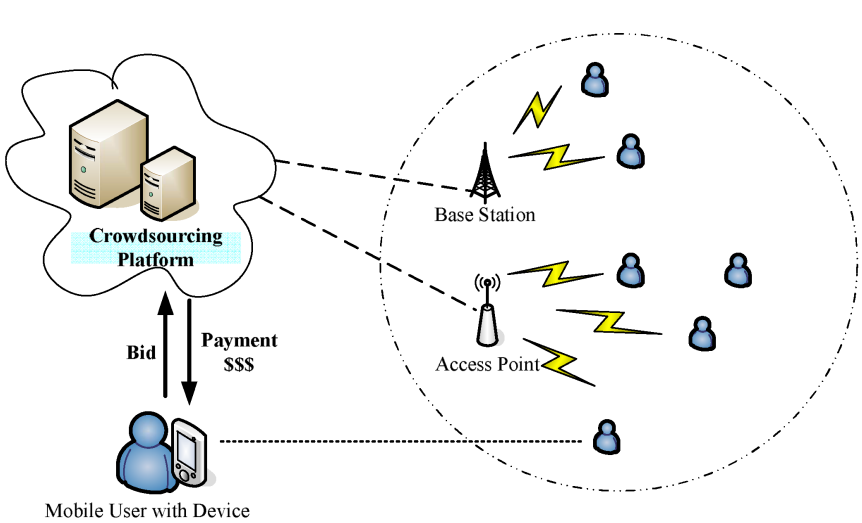
Outline

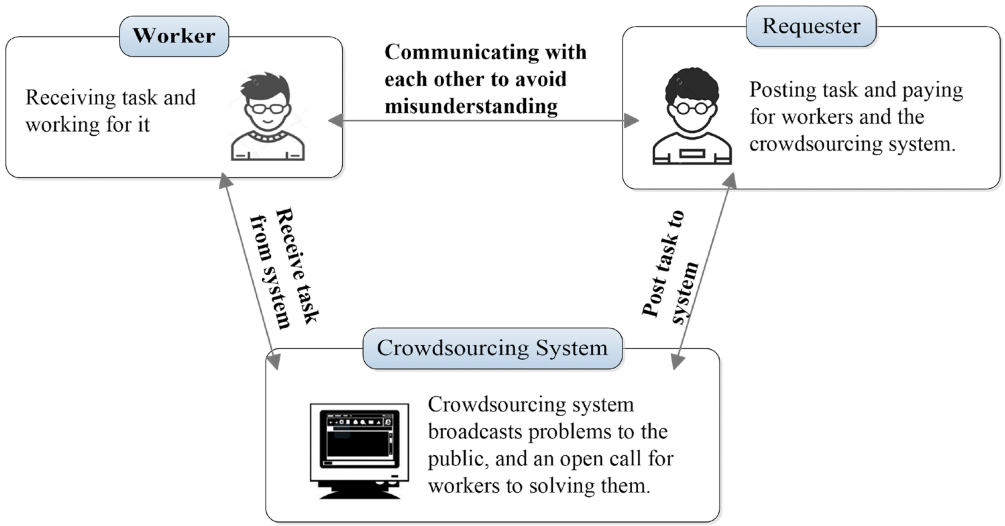
What is Crowdsourcing?

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 bt 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. **Rotalties from platforms**
   Most of crowdsourcing systems demands a sliding services fee for $5\% \sim 20\%$.

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?

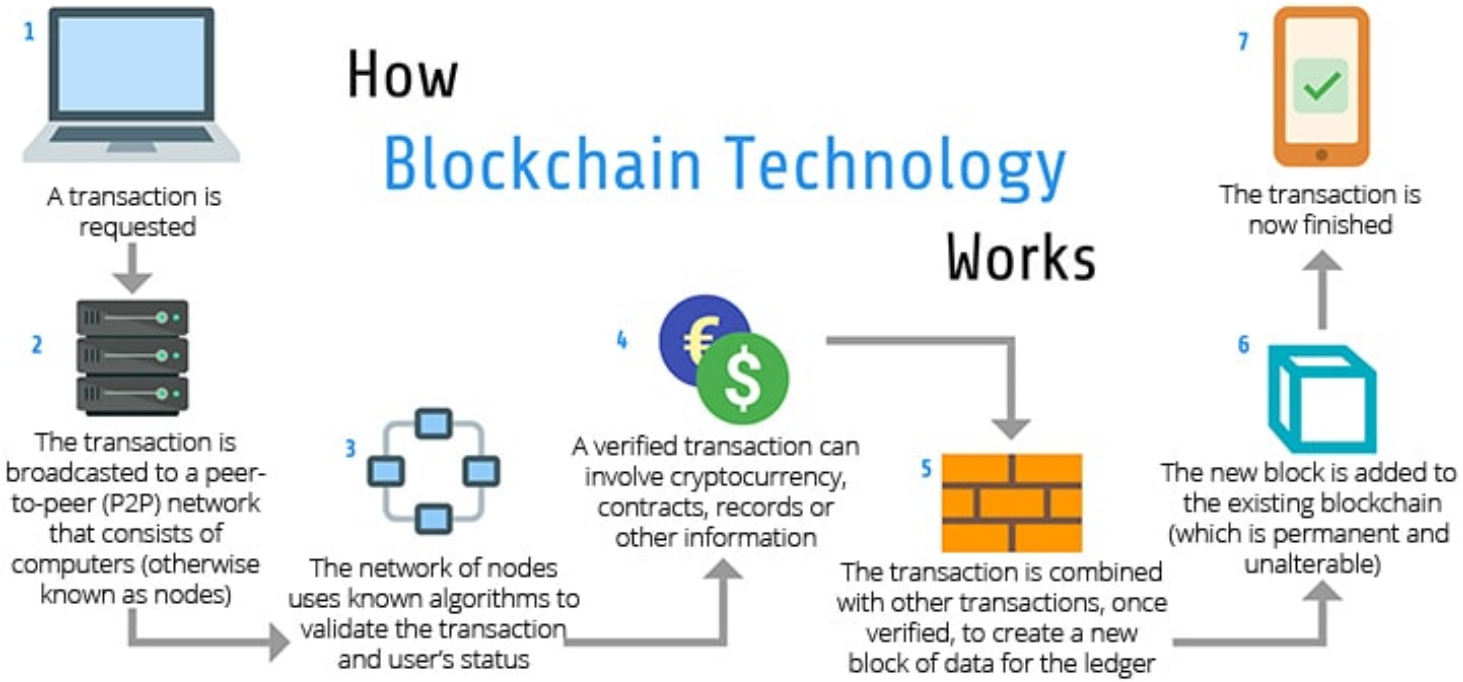
Let us embrace the future of Blockchain

Outline

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**.

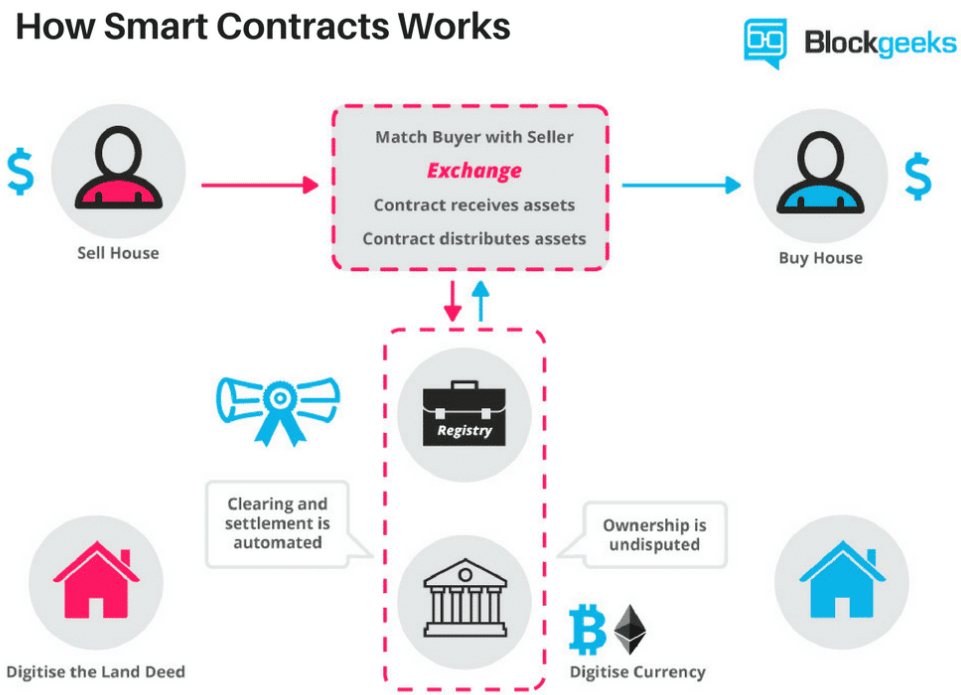


What is Smart Contract?

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

Four Roles: Requester, Worker, CrowdBC Client, Miner

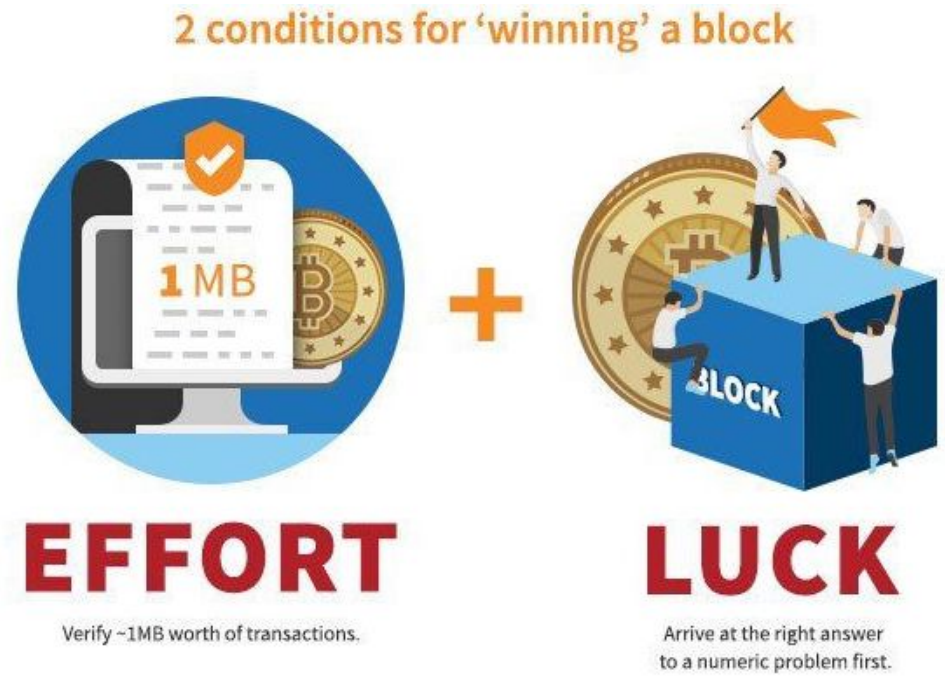
1. **Requesters**
   1. Identified by $\mathcal{R} \triangleq \{R_1, ..., R_n\}$, post tasks by providing descriptions
   2. Deposite (reward + monetary penalty) cannot be redeemed before ddl
2. **Workers**
   1. Identified by $\mathcal{W} \triangleq \{W_1, ..., W_m\}$, compete for tasks to get rewards
   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 β_{W_j}, W_j submitt solutions to $\mathcal{C}_k$ for ϵ_k times and got ψ_k high evaluation
   4. A deposit π_{W_j} is required to thwart DDoS and Sybil attack (How?)
   5. The solutions submitted by workers are identified by $\boldsymbol{s} \triangleq \{s_1, ..., s_m\}$

Four Roles: Requester, Worker, CrowdBC Client, Miner

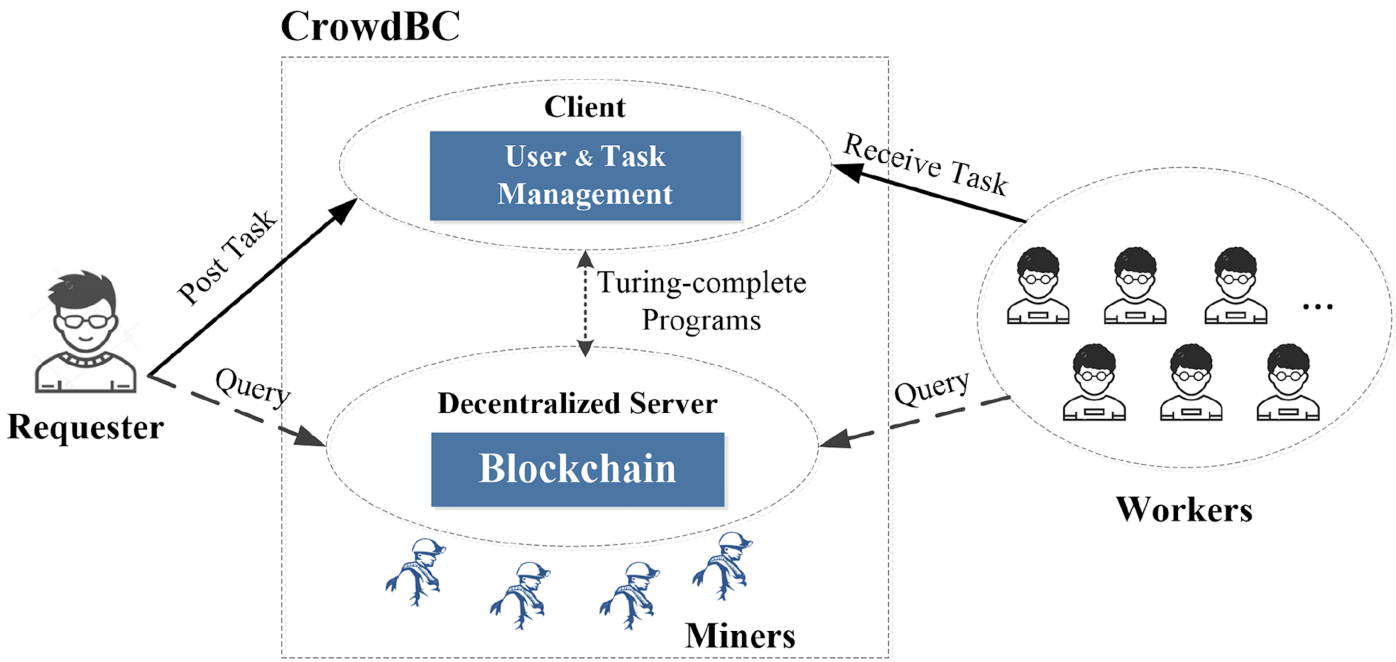
3. **CrowdBC Client**
   1. Served as a medium for $\mathcal{R}$ and $\mathcal{W}$, is not controlled by any third party
   2. Run locally on users' personal computer
4. **Miners**
   1. Add past transactions into a block and provide a valid proof (through mining) to claim the block reward and transaction fees
   2. $\mathcal{R}$ and $\mathcal{W}$ can also be miners if they participate in the mining



System Model Overview

Scheme

▷ $\mathcal{R}$ and $\mathcal{W}$ firstly need to register: $(K^p_{R_i}, K^s_{R_i}, K^a_{R_i}), (K^p_{R_i}, K^s_{R_i}, K^a_{R_i})$.
▷ 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

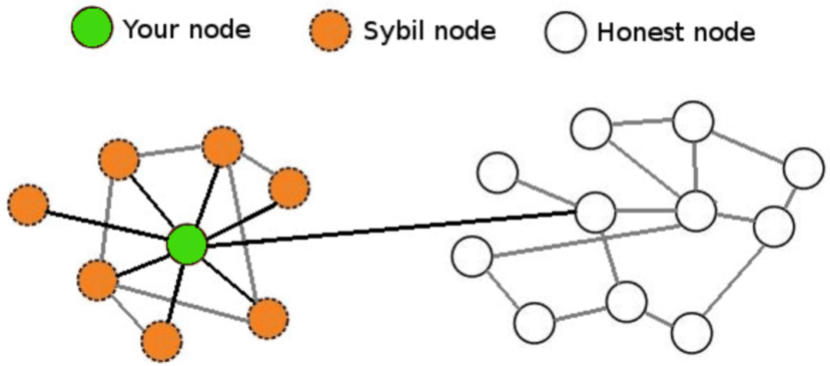


Potential Threats

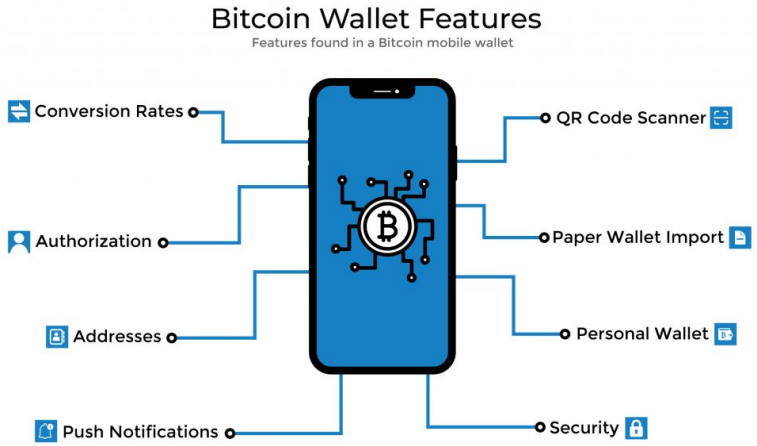
1. **Malicious Requesters**
   1. Collect usefull solutions without losing deposit **[false-reporting]**
   2. Misreport the high-quality solution as low level
   3. Deny the obtain of solutions
   4. Create a fork chain after receicing the solutions
2. **Malicious Workers**
   1. Obtain rewards without paying sufficient effort **[free-riding attack]**
   2. Create fork chain when receiving low level evaluation
   3. receive tasks but not submitt 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

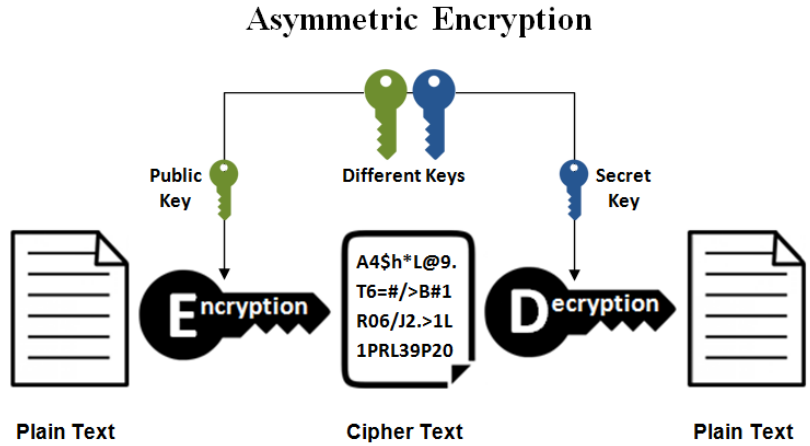
Security Assumption

1. **Majority Honest Security**



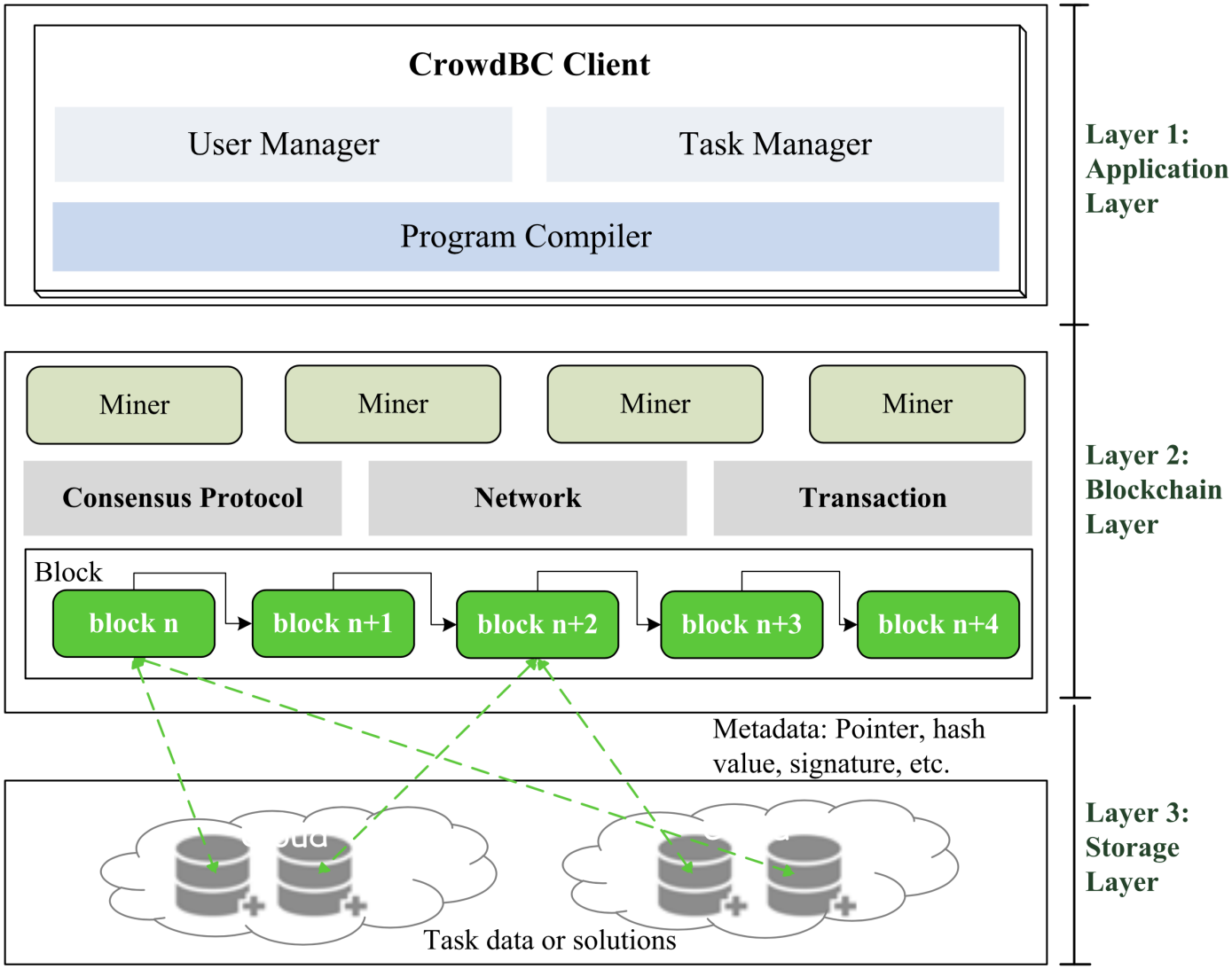
2. **Secure Transfer with Wallet**
3. **Secure Encryption Algorithm**





Outline

Three Layers Architecture of CrowdBC

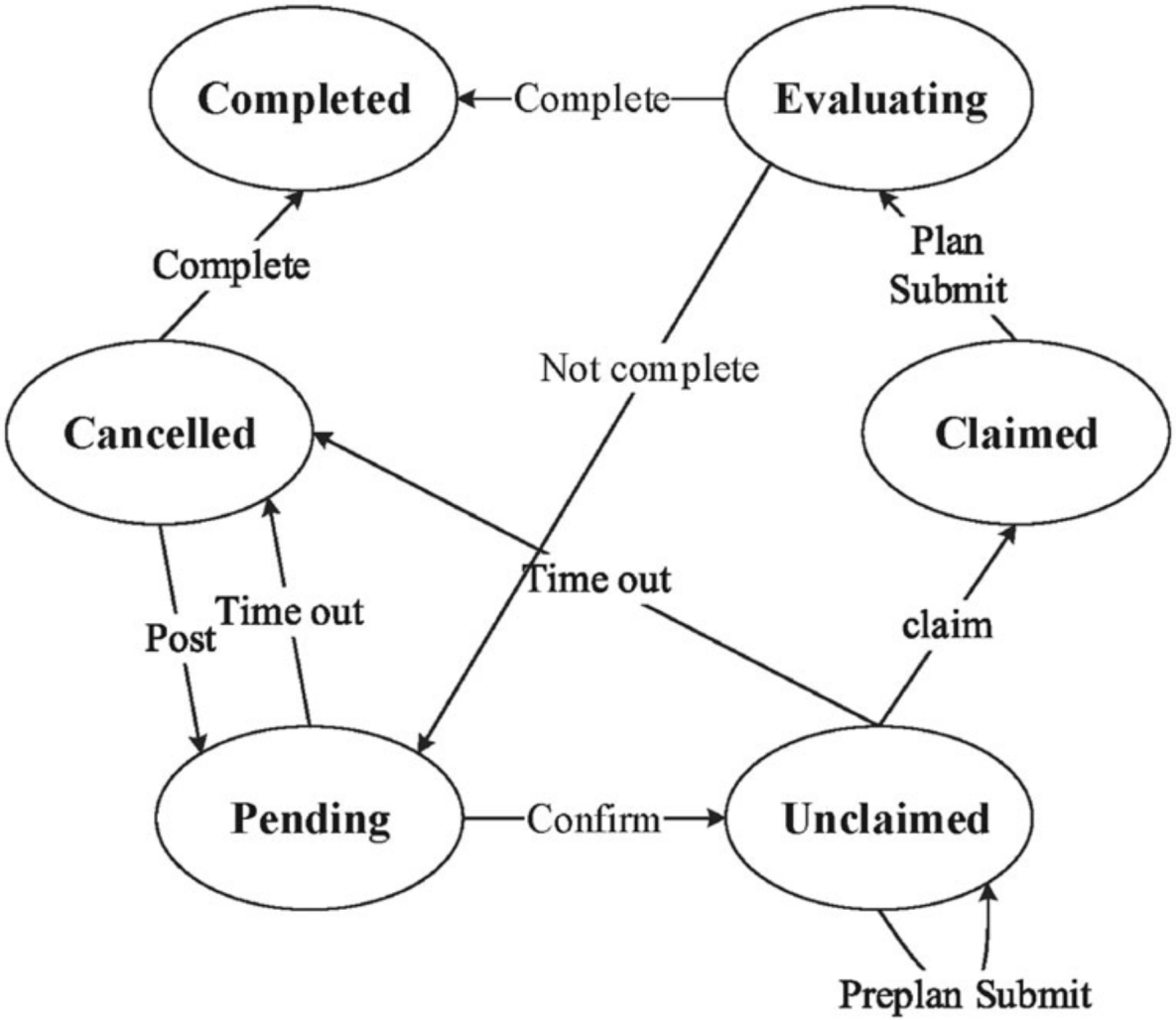


Three Layers Architecture of CrowdBC

1. **Application Layer [CrowdBC Client]**
   1. The entrance for users to finish a crowdsouring 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.

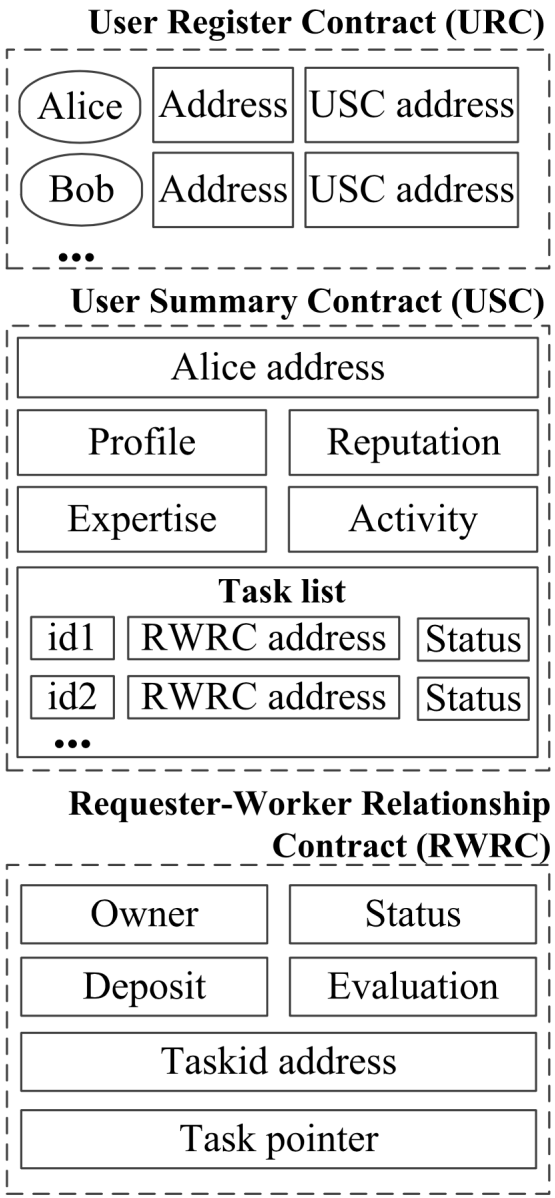


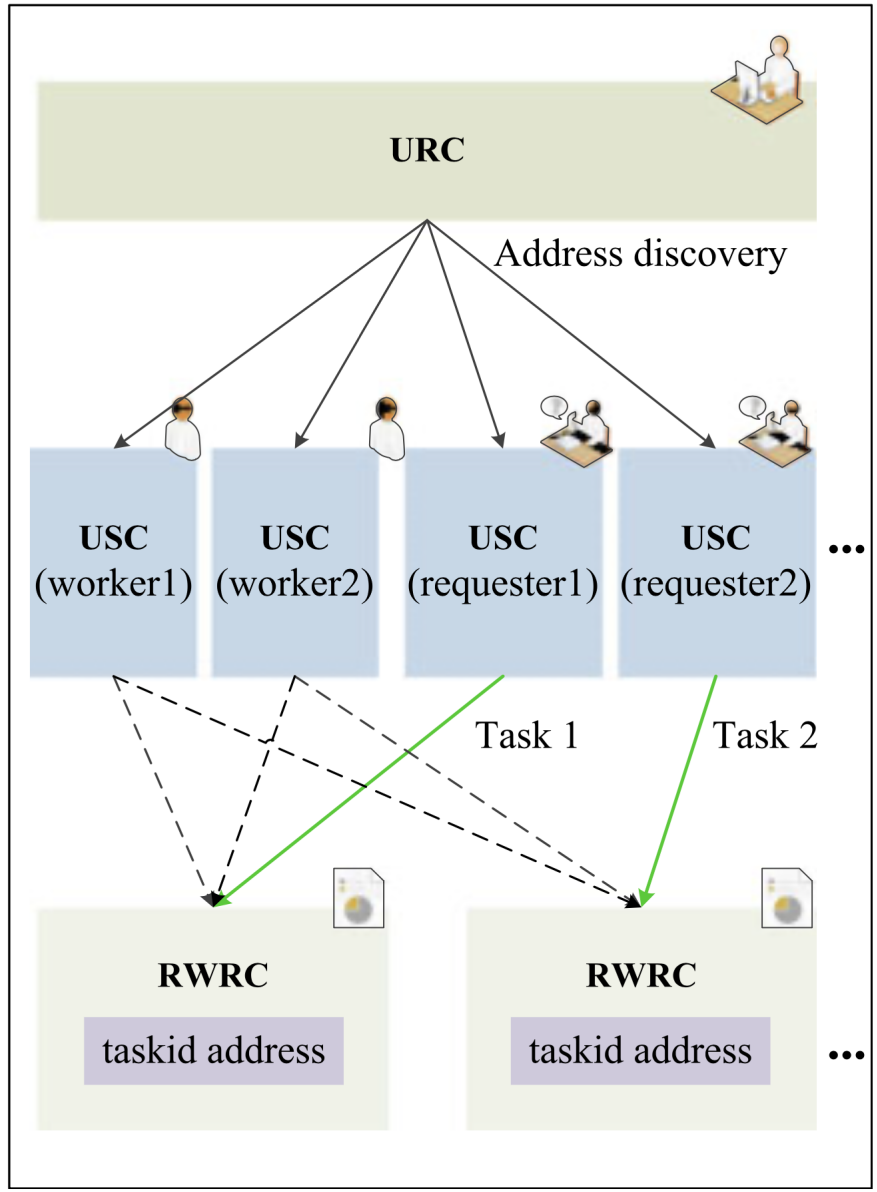
Crowdsourcing Process in CrowdBC

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** submitt 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

Designed Smart Contracts: An Overview





Designed Smart Contracts

1. **User Register Contract (URC)**
   1. Requesters & Workers do not need to submitt 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 τ_{ddl} or the task is received by some worker.

Phase #2: WDeposit. Before τ_{ddl}, W_j check if the remaining deposit is not redeemed and recevie the task if qualified.

Phase #3: Claim. Before τ_{ddl}, W_j submitts a solution to storage layer and give the addr to R_i. After confirmed by R_i, W_j submitts 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