

WorkerRep: Building Trust on Crowdsourcing Platform Using Blockchain

Student Name: Gurpriya Kaur Bhatia

IIIT-D-MTech-CS-GEN-18-MT16021

July, 2018

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee

Dr. Ponnurangam Kumaraguru (Chair)

Dr. Alpana Dubey (Co - Chair)

Dr. Arun Balaji Buduru

Mr. Vikrant Kaulgud

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science,
in General Category

©2018 IIIT-D-MTech-CS-GEN-18-MT16021

All rights reserved

Keywords: Crowdsourcing, Reputation, Blockchain, Ethereum

Certificate

This is to certify that the thesis titled "**WorkerRep: Building Trust on Crowdsourcing Platform Using Blockchain**" submitted by **Gurpriya Kaur Bhatia** for the partial fulfillment of the requirements for the degree of *Master of Technology in Computer Science & Engineering* is a record of the bonafide work carried out by her under our guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Ponnurangam Kumarguru

Indraprastha Institute of Information Technology, New Delhi

Abstract

Crowdsourcing is a process where an individual or an organization utilizes the talent pool present over the Internet to accomplish their task(s). These platforms offer numerous advantages such as reduced cost, better quality, and lower task completion time. To execute tasks efficiently, with the worker pool available on the platform, task posters rely on the reputation managed and maintained by the platform. Usually, reputation management system works on ratings provided by the task posters. Such reputation systems are susceptible to several attacks as users or the platform owners, with malicious intents, can jeopardize the reputation system with fake reputations. A blockchain based approach for managing various crowdsourcing steps provides a promising direction to manage reputation system. We propose a crowdsourcing platform where each step of crowdsourcing process is managed as transactions in Blockchain. This helps in establishing better trust in the platform users and addresses various attacks which are possible on a centralized crowdsourcing platform. We have built the proposed platform on the Ethereum framework. Our system utilizes IOTA's consensus mechanism which reduces the cost for task evaluation to almost zero. The cost incurred for complete cycle that is from posting task to updating reputation is \$ 0.631. Where in the cost for task poster comes out to be \$0.195 and for the worker its \$ 0.436.

Acknowledgments

I would like to express my deepest gratitude to my advisor Dr. Ponnurangam Kumaraguru for his guidance and support. The quality of this work would not have been nearly as high without his well-appreciated advice. I am especially grateful to Dr. Alpana Dubey for collaborating with us on this work and providing guidance. I would like to thank my esteemed committee members Dr. Arun Balaji Buduru and Dr. Vikrant Kaulgud for agreeing to evaluate my thesis work and for enriching this thesis with their valuable suggestions.

I thank all the members of Precog research group at IIIT-Delhi who have consistently helped me with their inputs and suggestions on the work especially Indira Sen and Simran Saxena. Special thanks to Shubham Gupta for helping me come up with the portal.

Last but not the least, I would like to thank all my supportive family and friends who encouraged me and kept me motivated throughout the thesis.

Contents

1	Research Motivation and Aim	1
1.1	Research motivation	1
1.2	Research aim	3
2	Related Work	4
2.1	Challenges in centralized crowdsourcing	4
2.2	Distributed reputation mechanism	4
2.3	Blockchain based crowdsourcing platform	5
3	Background	6
3.1	Crowdsourcing	6
3.2	Blockchain	7
3.3	Ethereum	9
3.4	IOTA	9
3.5	Distributed storage	10
4	Our Model	11
4.1	Terminology	11
4.2	Layout	12
5	Process	14

5.1	User registration	14
5.2	Post task	15
5.3	Task search	15
5.4	Task registration	15
5.5	Worker selection and task assignment	16
5.6	Task acceptance	16
5.7	Submission	17
5.8	Evaluation	18
6	Computing Reputation and Assigning Reward	21
6.1	Completing task	21
6.2	Evaluating submission	22
6.3	Updating reputation	23
6.4	Reward	23
7	Experiments and Results	25
8	Analysis of the Design	28
8.1	Privacy and anonymity	28
8.2	Decentralization	28
8.3	Robustness against various attacks	28
8.4	Case study of worker colluding with evaluators	29
9	WorkerRep:Portal	31
10	Conclusions, Limitations, Future Work	32
10.1	Conclusions	32
10.2	Limitation	32

10.3 Future work 32

List of Figures

- 3.1 Entities and various steps of crowdsourcing platform 6

- 4.1 Architecture of proposed crowdsourcing platform 12

- 5.1 Steps followed during registration process 14
- 5.2 Steps followed by task poster to post a task 15
- 5.3 Steps followed by worker to register for a task 16
- 5.4 Steps to perform assignment and acceptance of task 17
- 5.5 Steps explaining submission and evaluation of submission 20

- 9.1 Blockchain based crowdsourcing platform 31

List of Tables

- 7.1 Gas costs of executing various functions of our smart contract 26
- 7.2 Cost comparison of various crowd sourcing platform 26

Chapter 1

Research Motivation and Aim

1.1 Research motivation

Majority of the existing crowdsourcing platform such as Upwork [4], Topcode [3], AMT [1], etc. follow a centralized management approach; that is, they have a central authority through which each of the activities such as posting task, updating reputation, etc. are managed. Here, central authority refers to the people or the organization managing the platform. The central authority charges a fee as a part of getting a task done over the platform. The central authority provides some level of guarantee of security and fairness. And platform users have this assumption while using the platform. However, this may not hold true [25]. In literature it has been shown that there can be attacks on central authority either by outsiders or insiders [25]. Moreover, there might be privacy concerns to the user upon how his/her information will be used. In some instances, user certification authorities have been shown to turn unreliable [52]. **Reward** is given to workers by the task poster upon successfully completing the task. This serves as an incentive for him/her to complete the task. Some crowdsourcing platforms require the task poster to deposit the reward amount before beginning with the tasks. This gives central authority an undue opportunity of deciding how it can use the money. The worst case, they may take away the money without user's consent [38] or may not reward the worker suitably on successful completion of the task.

Even if we assume that central authority is trustworthy, there is a need to establish trust among the task poster and the worker for the success of the platform [20]. Trust among users on platforms can be built by using ratings given by the task posters and workers which is further used for computing reputation score for individual users [36]. Majority of existing studies have approached this from the perspective of centralized crowdsourcing platform [29, 43]. On such platforms, integrity and authenticity of the feedback received and reputation scores computed cannot be guaranteed as all the control resides with the central authority and it is easy for them to manipulate it. We propose a distributed and decentralized crowdsourcing platform which

addresses the issue of complete reliance on central authority for the reputation management.

1.2 Research aim

In particular, the contribution of this thesis is:

- **Decentralized crowdsourcing platform.** We model crowdsourcing platform with blockchain based design principles where all the important steps involved in crowdsourcing process are captured in the form of transactions on a blockchain network. The transactions are hashed cryptographically, validated based on consensus, stored by multiple entities subscribed to the system and are immutable.
- **Reputation mechanism.** By designing an evaluation strategy that is insusceptible to malicious users, we build a robust and immutable reputation system for our platform.

Chapter 2

Related Work

2.1 Challenges in centralized crowdsourcing

With the rise in conventional workforce moving towards gig-economy [7] there has been widespread conceptualization of it in different areas social science and humanities [23] to software engineering [32].

Various challenges of centralized crowdsourcing platform such as trust management [20, 21, 27, 43, 44, 46, 49], incentive mechanism [22, 28, 46, 53], quality control [10, 13, 34, 51], privacy and security [40, 41, 48] have been taken up in literature.

2.2 Distributed reputation mechanism

Work on distributed reputation system by [11] where they have presented a privacy preserving reputation mechanism by using a pseudonym for users rather than their actual identity. They have also integrated a certification authority for authentication of the users on their platform. It also keeps discarding old reviews and provides no mechanism to prevent or handle unfair rating. [50] Proposes to use the most recent rating received by the user from the rater to overcome collusion attack, but this might not be reflective of its past behavior and it would be easy to manipulate the rating. [16, 18, 37] Propose reputation systems that leverages blockchain. [16] associates fees with every feedback the seller wants. This can reduce reviews that are given for transactions that have not been performed but does not eliminate it. Also it increases the cost involved in selling an item for the seller. And might be burdensome for the seller. Authors in [37] have focused on computing the reputation of the seller on an e-commerce platform using reviews (rating as well as textual) that he/she received. Each review is bound to a transaction and reviewer anonymity can only be guaranteed if there are a number of transactions that are to be reviewed which involves the seller in a given time. Ways of overcoming biased reviews have not been proposed. Dennis and et al [18] use IP addresses to bind the identity of the user on their platform. Spoofing IP addresses is fairly easy now. In order to avoid collusion attack they

are taking an average of the scores received by the worker. Since averaging is sensitive towards outliers, it might reduce the effect of collusion but does not eliminate it.

2.3 Blockchain based crowdsourcing platform

Authors in [30] have proposed a crowdsourcing platform built on Ethereum and task assignment is done on first come first serve basis which might result in poor quality submissions from the assigned worker. Submissions are evaluated within the smart contract using an evaluation function provided by the task poster. They have proposed a reputation management scheme but have not looked upon various attacks that can occur. The other work that is similar to ours is of [31]. They have proposed a crowdsourcing platform leveraging blockchain and that largely focuses on preserving privacy and anonymity of their users. However they have not presented a reputation mechanism for their system and require users to have a new account for every transaction on the system. Both the above implementations require central authority for authenticating user identity and issuing pseudonyms to the users.

We propose a crowdsourcing platform that is decentralized and does not rely on third party for its core functionalities and it is built on top of Ethereum. It also has an added advantage of reduced cost as the task poster is not required to pay to any central agent to get his/her task done. We also try to prevent some of the various attacks on the reputation system, mentioned above, by building a stronger submission evaluation methodology.

Chapter 3

Background

3.1 Crowdsourcing

Jeff Howe first coined the term crowdsourcing, describing it as "the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined network of people in the form of an open call" [6].

Several crowdsourcing platforms have emerged in past decade such as Amazon Mechanical Turk [1], Upwork [4], Topcoder [3], etc. These platforms offer numerous advantages such as reduced cost, better quality, and lower task completion time. Because of these advantages, tasks as simple as data annotation to as complex as software development are being crowdsourced.

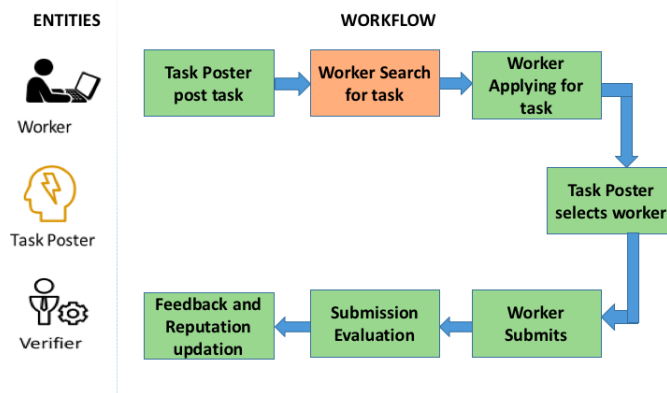


Figure 3.1: Entities and various steps of crowdsourcing platform

In general, a crowdsourcing platform constitutes of three entities, namely the **task poster** who needs to get a task completed, the **worker** who takes up the task to complete it and the **platform** that provides a medium to task poster to post the task and workers to search and submit the task. There are two types of crowdsourcing platforms: (1) **hiring based** [4], in which the workers apply for a task. After the application process is over the task poster chooses workers from the set of applicants, to work on the task. The number of workers depends upon the requirement of the

task poster. Only the selected workers are eligible for the reward, upon successful completion. The entities involved along with the basic work flow followed by the platform is shown in Figure 3.1. (2) **Competition based platform** [3], are those where there is no specific worker selection phase initially. Any number of workers can compete for the task completion. Reward for the task is decided by the task poster which is usually based on the quality of completed task. In this thesis we focus on hiring based crowdsourcing platform whose work flow is as follows. The task poster posts the task over the platform, workers search for the most suitable task and register for it. The task poster selects the worker who will be working on that task and the worker accepts the offer and starts working on it. After completing the task the worker submits it to the task poster it is then evaluated by the task poster or the evaluators on the platform and the worker is rewarded suitably.

3.2 Blockchain

Blockchain, invented by a person or group of people known by the pseudonym, Satoshi Nakamoto, is a method to maintain a distributed ledger. The ledger is a growing list of records clubbed into a block. It contains a list of all valid transactions that happen between different nodes over the network. In blockchain network, each node maintains a copy of the ledger. Whenever a new transaction happens, it is first verified by the miners before being added to the ledger. Miners are node in the network that validate transactions occurring in the network. The protocol followed in validating a transaction is designed in such a way that it avoids attacks such as double spending, balance insufficiency, etc. The blocks are mined using the proof of work concept, which is solving a computationally intensive cryptographic puzzle using hit and trial. The cryptographic puzzle is finding a hash value for that block which is equal to or lower than the target at that point. A target is a 256-bit number that approximates the difficulty of the network. The difficulty here refers to the difficulty in mining a block, that is to find the acceptable hash of the block. Bitcoin [33] which is the first digital cryptocurrency, uses SHA256 hashing algorithm to hash the blocks. The input to the hashing algorithm is the version number of the block, the hash of the previous block stored in the header of the newly created block, Merkle root of the transactions in the block, current time stamp, difficulty, nonce which is a 32-bit number. Nonce helps generate different hash value, and if the nonce overflows, an extraNonce field in the coin base transaction(leftmost leaf node of the Merkle tree) is incremented changing the Merkle root hence generating a different hash with already tested nonce value. Storing the hash of the previous block helps form a linked chain making the blockchain immutable and more trust worthy since altering any transaction inside the block would require recomputing the hash of the current block and all the block mined after the current block. But it is not impossible to break the network, that is, have control over the blockchain network and manipulate the ledger. It can be done by having the control over more than half of the computational power of the network, referred to as 51 percent attack [30]. Besides the well-known application of blockchain which is the crypto currencies, there are other areas where it can be advantageous to apply the concepts of blockchain such as management

of authorship and ownership [14,15], identity management [42], E-voting [26, 35, 45], protecting personal data [14], decentralized governance [54], managing IoT devices [12].

3.3 Ethereum

Ethereum, which is an extension to the Bitcoin blockchain, contains an added feature known as the smart contract. It serves the purpose of physical (paper) contract, but the only difference is that they are digital. Each smart contract is represented by a piece of code known as the contract code. This allows automatic execution of the contract code upon meeting certain condition(trigger). The contract code can also contain various functions that are invoked by the user. The execution of the code is represented as a transaction on the Ethereum network. All such transactions are irreversible, and no third party is involved as a middleman to account for the credibility of the parties involved in the contract. This builds a superior form of trust and security for the people involved in the contract. As only what has been specified in the contract will take place, no third person/party can alter it, not even them. The mining architecture of the Ethereum network is similar to bitcoin network; the only difference comes while verifying the transaction. In the bitcoin network, miners verify the transaction by cross-checking with the previously carried out transaction on the network, whereas in Ethereum network, the miners verify the transaction by running the code specified in that transaction. On the Ethereum network, ether(ETH) act as a currency or an asset that can be used to pay to the Ethereum network nodes to get some actions executed. **Gas** is the execution fee which measures the computational complexity of the action that is to be executed. The number of Ethers to be paid for an action depends upon the gas required. This is done to keep the measure constant since the value of Ether keeps fluctuating.

3.4 IOTA

It is similar to the blockchain, but there are two main differences in the architecture part. First is the structure and the second is the consensus. In blockchain, miners now become pretty separate from the people doing actual transactions. That is there are mining pools now that perform the mining. So as these mining pools are increasing in size, the system is becoming somewhat centralized. In this new concept of IOTA, if a node has to perform a transaction he is required to mine two transactions from the pool of unconfirmed transactions instead of paying transaction fees. The other advantage is that there isn't a scope of the system to become centralized since each node carrying out a transaction is a miner now. Thirdly, in blockchain network with an increase in the number of nodes in the network, it becomes slow. Due to the fact that the number of nodes that mine the transaction is generally constant but the nodes carrying out the transactions increase in number. This increases the number of unconfirmed transaction at a given point of time, whereas in IOTA as the number of nodes increases the scalability increases as there are more nodes to mine the transactions.

3.5 Distributed storage

Storing information on blockchain is very expensive [47] as compared to storing it to external storage. If the external storage is centralized it will suffer from privacy issues, security issues, etc. as said above for centralized system. So using a distributed storage such as IPFS [15], or swarm [24] over a centralized storage helps us overcome the shortcoming in a similar way as the blockchain based platform does for a centralized counterparts. IPFS or swarm are both peer to peer storage systems. Upon storing the data in them, a hash of the data is returned. This hash is used to access that data by anybody, anytime in future. Once a data is stored on them, it stays there till the time even a single node in the storage network has a copy of that data. If we want to access particular data in IPFS storage, we ask in the network which node has the hash (hash here refers to the hash of the data). If any of the nodes have it, they return the data to the one asking for it. In case of swarm, the hash is the address where the data is stored. So querying the system with the hash directs to where the data is stored, that is it acts as an address for the data. We will be using IPFS as our external storage in rest of the paper.

Chapter 4

Our Model

We propose a distributed blockchain based crowdsourcing platform which models a hiring based crowdsourcing. There are mainly two entities on our platform (1) the task poster and (2) the worker. Unlike most of the crowdsourcing platforms, where a task is evaluated by task poster, in WorkerRep tasks are evaluated by peer workers present on the platform. Based on their skills and reputation on the platform, they are pseudo-randomly selected for task evaluation. Here, task evaluation implies judging if the completed task satisfies the task requirement.

4.1 Terminology

In this section, we describe various attacks and other terminologies used in the thesis.

- *Reputation* is the measure of how well the worker has performed in the tasks that were previously assigned to him and how well he evaluates the work done by others.
- *Initialization and Cold Start Problem* is a problem faced by the new users on the platform due to low reputation score. [39].
- *Sybil attack* is an attack where a malicious worker tries to create multiple identities over the platform to gain influence on the platform [18]. Generally, it is done to carry out some of the below-mentioned attacks:
 - *Re-entry attack* carried out by creating a new identity on the platform, leaving an identity with a bad reputation. Generally, reputation lower than what is for a new worker [39].
 - *Collusion attack* is when a group of workers try to collude together to improve their own reputation or decrease the reputation of others [39].
 - *Ballot Stuffing* is when the worker tries to increase its own reputation. [18].

- *Unfair rating* attack is when the rater is biased towards worker and does not give a truthful opinion about him. If its biased in a negative sense, that is rater tries to decrease the reputation of the worker it is known as bad-mouthing.
- *Reciprocity* is when worker reciprocates negatively for a negative review that he receives [39].
- *Whitewashing* attack happens when either the worker knows how to manipulate the reputation system or by re-entering the system [25].

4.2 Layout

Since our proposed system is built on top of Ethereum network, smart contract is the most fundamental building block for it. Our system has five kinds of smart contracts: (1) *UserContract*: contains functionalities to create new users on the platform. (2) *TaskContract*: it offers functionalities to create a new tasks as well as view about the existing tasks on the platform. (3) *AgreementContract*: creates an agreement between the task poster and the worker corresponding to a given task. (4) *SubmissionContract*: the worker invokes the functionalities of this contract when he/she has finished the task he/she was assigned and is ready to submit. The basic functionality of this contract is to accept submission from the worker and the assign evaluators for that submission. (5) *EvaluationContract*: provides various functionalities to the evaluators of the tasks. It also computes and updates the reputation of workers based on the evaluation score received from the evaluators. The architecture of the system is shown in figure 5.4.

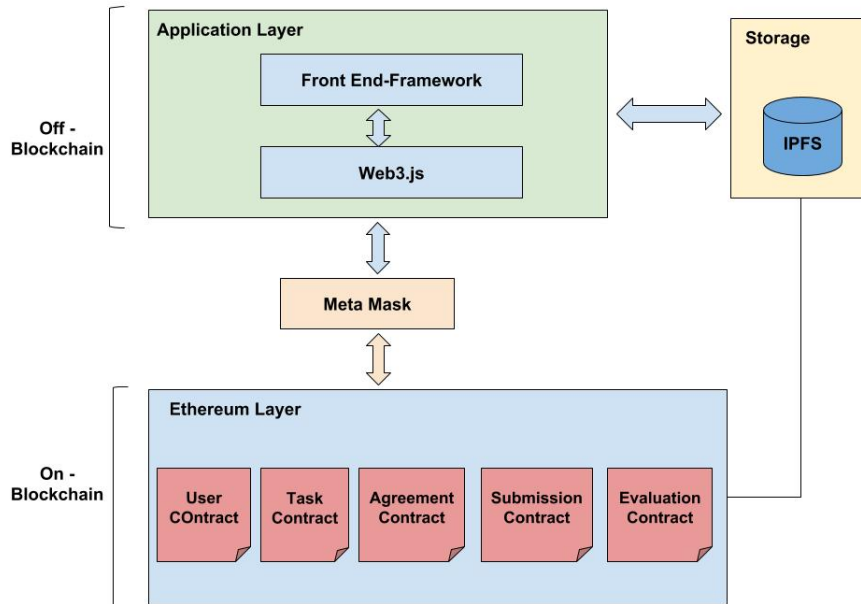


Figure 4.1: Architecture of proposed crowdsourcing platform

To perform any action on the platform the user has to call the function of the contract corresponding to that action. If the function call results in a change of state of Ethereum Blockchain then the function call is treated as a transaction from the one calling that function. Transactions on Ethereum network are cryptographically signed using an asymmetric encryption algorithm to prevent non-repudiation of the origin of the transaction as well as to maintain the integrity of the data. Miners on the other hand, can decrypt the transaction using the initiator's public key. After decrypting it they verify the validity of the transaction by referring to the transactions sent and received by the sender. If found valid, these transactions are appended to the public ledger. These transactions cost Ethers to the users. Interactions with the Ethereum network are carried out by using **MetaMask** [2]. MetaMask is an application that acts as a bridge between the browser and Ethereum. In figure 3.1, function calls corresponding to steps colored in green are treated as transactions and carried out on Ethereum blockchain and the ones in orange that is task search and apply for a task do not involve appending transaction to the Ethereum blockchain.

Chapter 5

Process

Sections below define how each of the steps mentioned in figure 3.1 are performed on our platform.

5.1 User registration

Users register on the platform as worker or task poster and specifies his/her public key and the IPFS. Ethereum wallet address is used to register on the platform. Each user is linked to a particular address and public-private key pair, the users created from different key pairs will be different. So to avoid Sybil attack, we ask the user to pay some amount of Ethers while registering. Figure 5.1 explains diagrammatically the various steps followed by the user while registering on the platform.

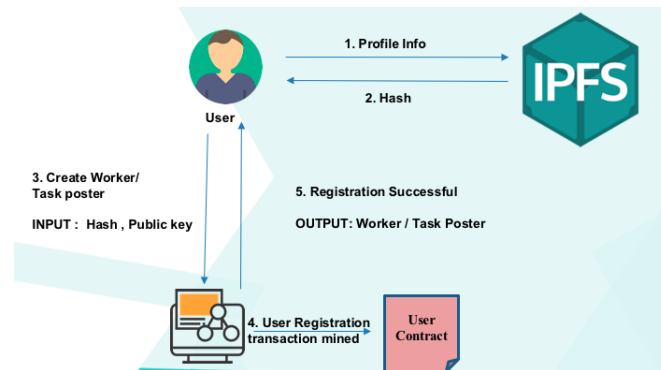


Figure 5.1: Steps followed during registration process

Authors [19] have shown that if there is a cost associated with generating identity, Sybil attack is greatly reduced. Another way to avoid Sybil attack is to associate identity with the user, but it requires a third party to validate the identity. We use the former approach of handling Sybil attack due to the following reasons. Firstly, including a third party contradicts our architecture of a decentralized system. Secondly, asking for a fee initially would keep spammers away from our system, and only those who are keen on posting task or performing a task will be joining the

platform. The Ethers taken as fee from the user are stored in their account on the platform and can be returned when the user leaves the platform. The amount returned to the user depends on the reputation the user has at that point in time.

5.2 Post task

To post a task, the task poster specifies the reward for the task, title, skills required and the IPFS hash of the task metadata and sends the transaction. Upon the transaction being mined, the task gets appended to the public ledger as a transaction carried out by the task poster and to the list of available tasks in the *TaskContract*. This step is incorporated within the Ethereum network so that further transactions happening for this task such as assigning the task to the worker, submission by the worker, etc., can be linked to it. It will help ease verification process carried out by other task posters before assigning a worker to their task. Figure 5.2 shows the various steps task poster needs to perform in order to post task on the platform.

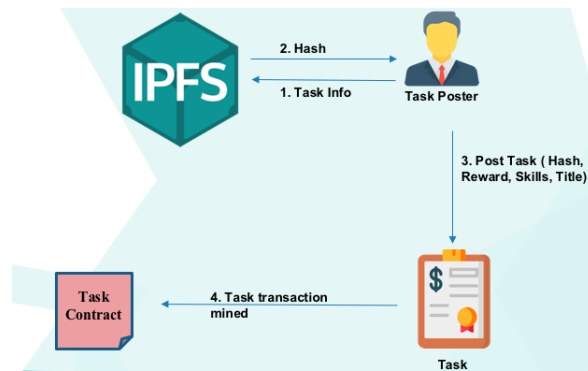


Figure 5.2: Steps followed by task poster to post a task

5.3 Task search

Once a task is posted, it is added to the list of available tasks for workers to perform. All the available tasks and their relevant details are public and available for view. Workers can apply filters based on skills, duration, etc on the collection of available tasks to aid them in choosing tasks.

5.4 Task registration

In this step, a worker can chose to apply to an available task posted on the platform. By applying, a worker can express his interest towards that task. Details of workers who have applied for the tasks are visible to task posters and these can help the task poster in choosing worker(s) for their task. Steps followed are graphically shown in Figure 5.3.

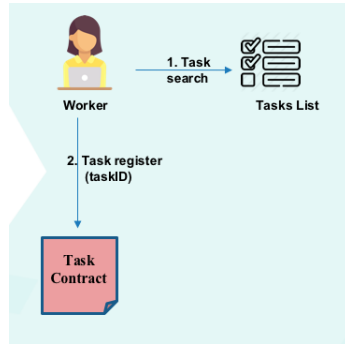


Figure 5.3: Steps followed by worker to register for a task

5.5 Worker selection and task assignment

Selecting the most suitable worker from the applicants is a time-consuming process. The task poster can initially filter out workers based on their reputation score if required. Once the task poster has selected worker/s for his/her task, he/she creates an agreement using the *AgreementContract* specifying the worker information within it. Agreement here is a binding structure between worker and task poster for a given task that is able to enforce the rules of the system associated with the task such as acceptance, submission, reward and the updates to reputation. While creating the agreement, the task poster is required to send the reward amount for that task, to this agreement. This amount can only be withdrawn if the worker has not accepted the task and the task poster wants to cancel the task or can be awarded to the worker upon completion of the task. Since there is no central authority involved the money is kept safe in the contract and will be awarded to either of the two entities involved in the agreement based on above mentioned condition. Letting task poster send in money while creating assignment will help in avoiding task posters to assign numerous task at a time, hence spamming the network and also ensures fair payment for the worker. After the agreement is successfully created, the selected worker is notified for him/her to accept it and start working on it. These transactions are recorded into the Ethereum.

5.6 Task acceptance

Once the task poster assigns the task to the worker by creating an agreement, the worker is notified of the same. To accept the task, the worker is required to send some fixed amount of Ethers to that agreement. The Ethers that are deposited by the worker will be refunded based on his/her performance. Upon receiving the Ethers from the worker the contract code will check whether the address of the sender of those Ethers is the same as worker address specified by the task poster while creating that agreement. If it is the same, the task poster will be notified of the acceptance and the worker can start working on the task else the gas corresponding to that transaction will be consumed. The task can be accepted by a specific date mentioned by the task

poster, post which the agreement will stand canceled automatically and all the Ethers residing in that agreement will be sent to the creator of that agreement, that is to the task poster. Post this date the worker will not be able to accept the agreement. This is done so that the task poster does not keep waiting for the worker to accept the agreement. If the worker does not accept the task, the task poster can claim the Ethers back and use them to assign the task to another worker. Just so the task poster does not explicitly kill the agreement in the middle, we keep a check if the worker has accepted the agreement. If he/she has paid, then the contract cannot be killed by the task poster, and it will only get terminated once the due date is reached. Collecting Ethers from the worker when he is accepting the task works as an assurance to the task poster that the worker would not leave the task halfway. If he does not complete the task, the Ethers submitted by the worker will be transferred to the task poster as a compensation. And if he does the submission, the fees returned will depend on the completeness of the submission.

5.4.

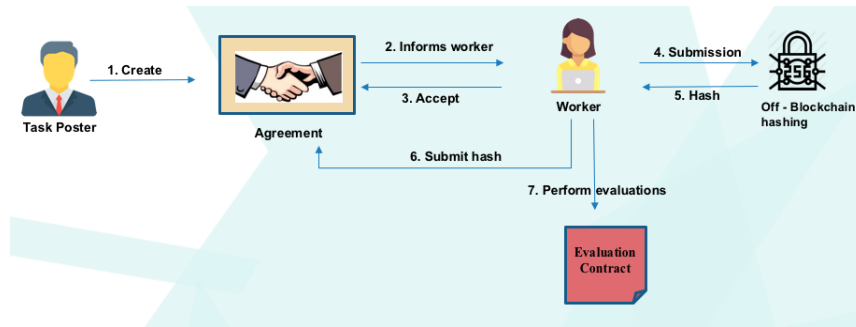


Figure 5.4: Steps to perform assignment and acceptance of task

5.7 Submission

After the worker has completed the task, he can submit the solution/submission by invoking the functions from the *SubmissionContract*. Initially the worker hashes the submission using the Keccak-256 hashing scheme as used by Ethereum and sends it to the agreement before the due date. Any action that is performed on Ethereum is publically visible and we do not want to make the submission public. Hence hashing performed by the worker is not part of the Ethereum network and is performed off blockchain. This hash received by the *AgreementContract* serves two purposes (1) it helps platform know that the worker has completed his/her work and is ready to get evaluated. (2) It helps prevent the worker from changing his submission upon knowing who his/her evaluator will be. So, upon receiving the hash, evaluators are pseudo-randomly chosen as discussed in the section below and their public keys are returned to the worker. Evaluators are not notified to the worker initially as it may introduce bias in his work if someone he knows becomes the evaluator of his task.

The worker then encrypts the submission for each of the assigned evaluator using their public key which is again encrypted by his/her own private key and is then stored on IPFS. The submission

is not stored on blockchain due to (1) cost involved to store large data on the blockchain network is high. (2) Privacy concern, the task poster might be reluctant to share the submission with the public. The hashes received by storing the encrypted submission for each individual evaluator are combined and then returned to the *AgreementContract* by the worker. First level of encryption provides confidentiality as nobody other than the evaluator having the corresponding private key can access the submission. Second level of encryption provides authentication of the worker. The case where evaluators explicitly share the submission with someone else can not be handled by our system. Encryption or decryption is not secure on blockchain since the keys become publically visible as in the case of hashing the submission as discussed above so, it is performed off blockchain at the user side. Storing the encrypted hash once for each worker consumes a lot of space. Instead, we can store the submission once over IPFS and perform the two-level encryption as described above to the hash received from the IPFS rather than performing it on the worker's submission itself. This would greatly reduce the space required on the IPFS as the number of evaluators increase but such a scheme can result in privacy and security issues as anybody with the hash can access the submission.

5.8 Evaluation

Evaluation on the existing crowdsourcing platforms is generally performed by the task poster itself. We propose a different model in which workers on the platform evaluate the submission of their peer workers and we also assume that the number of workers actively participating on the platform are sufficiently high. A worker evaluates a task when (1) the worker wants to get his submission evaluated and (2) the worker is willing to evaluate the submission given he/she has sufficiently high reputation. Our design requires the worker to evaluate y other submissions before his submission gets evaluated . Here y can depend on various factors such as the difficulty of the task or "task evaluation", number of task that are yet to be evaluated, number of workers who are ready to evaluate and reputation of the worker. One advantage of using this kind of evaluation model is that, as the number of users increases in the network the efficiency of submission evaluation increases. The above evaluation model has been borrowed from IOTA's consensus model [5]. In first case, various incentives for the worker to perform evaluation are that he does not have to pay fees to get his submission evaluated, he might learn new things that could be helpful for him in future and there is an increase in reputation upon correct evaluation of peer's submission. In the second case, incentive to evaluate others submission is, to get their reputation increased and to learn. An increased reputation score in the latter case can easily be abused by a worker with a lower reputation as they would find this as a swift option to boost their reputation. To prevent this, a worker who has no submission that is pending for evaluation and has a low reputation score is not allowed to evaluate others' submissions. Only workers whose reputation is above a threshold are allowed to evaluate task if they have no pending submission to be evaluated. Further this threshold can be altered dynamically by the platform to adjust to the current requirement of the number of evaluators.

There are two parts of review given by each evaluator (1) Score, which is provided based on completeness and quality of the submitted task. Evaluator rates the submission a value between [1, 100] on both the criteria. One means poor performance and 100 means excellent performance. A range from 1 to 100 has been chosen to overcome the technical limitation of solidity which is the programming language for creating smart contracts. Such a range provide balance between precision and memory requirements. Ideally this range can be anything. (2) Textual review, a brief explanation of the rating given by him/her.

Since peer workers are evaluating the task, there are chances that the evaluation may not be a true reflection of the work done by the worker due to unfair rating or collusion between the worker and the evaluators. To avoid it, we assign a total of x evaluators to each task and take consensus about their evaluation score inorder to provide the worker with a fair evaluation. The incentive of increased reputation score as mentioned above might prompt the evaluator to randomly assign evaluation score to a submission. To decrease the effect of such random evaluation and unfair rating, consensus among the evaluators is taken and the incentive of increased reputation is received by only those evaluators that are part of the consensus. Consensus mechanism is explained in Chapter 6.

The number of evaluators for a given submission can depend on various factors such as the complexity of the task which can be measured in terms of the duration required to complete the task, number of workers available to evaluate the task which further depends on the number of workers with the required skill and reputation, the number of workers who have to get their submission evaluated and the number of workers having the required reputation and willing to evaluate the submission. Once the number of evaluators is known, we next consider how they are chosen. For a given task, workers having the skill to evaluate that task are divided into slots based on their reputation score. The number of slots depends on the number of evaluators required for that task. Evaluators are then randomly chosen, one from each slot, assuming there always exists at least one worker in all the slots. The advantages of using this method are (1) allowing an evenly distributed pool of evaluators from different reputation scores to get better evaluations. (2) prevents workers who have made a submission and have a low reputation score from starving due to them not being allocated a task for evaluation, resulting in their task not being evaluated. (3) reduced probability of colluding workers being chosen together to evaluate the same task as a successful collusion will require the colluding workers to firstly be in the list of eligible evaluators. Secondly, they would be required to be distributed in the reputation slots in such a way that would increase the chances of one of them getting selected from at least half of the slots. The collective effort and finances required to arrive at such a state far exceed the benefits that could be derived from it. Figure 5.5 shows various steps followed in order to perform submission by the work and evaluation of that submission. 5.5.

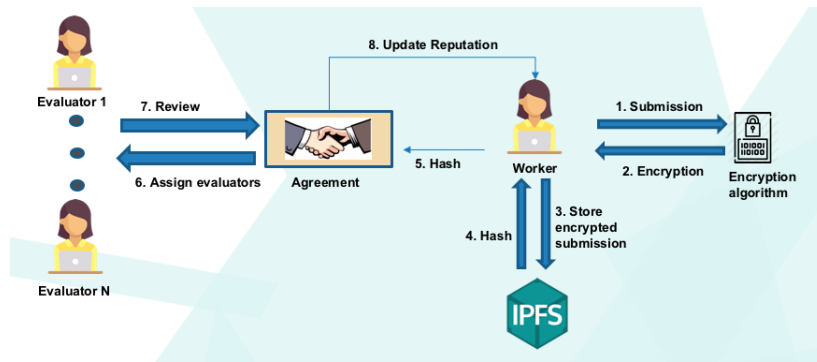


Figure 5.5: Steps explaining submission and evaluation of submission

Chapter 6

Computing Reputation and Assigning Reward

There are two ways through which the reputation of the worker gets updated (1) by completing task and (2) by evaluating others submission. Initially the reputation of the worker is set to 1. Performance of the worker in every task that he/she has completed or evaluated, adds to their reputation on the platform. The reputation earned by the worker for on his/her submission is dependent on the credibility of the evaluators and the score he/she receives for his/her submission. Likewise, when he/she evaluates a submission, then reputation depends on how close his/her score is to the score assigned to the worker.

6.1 Completing task

As mentioned in chapter 5 there are two criteria based on which the evaluators score the worker, that is completeness and quality. Let e_1, e_2, \dots, e_n be the set of evaluators chosen pseudo randomly and c_{ik} and q_{ik} be the *completenessScore* and *qualityScore* respectively given by the i^{th} evaluator for k^{th} submission. The mean of the *completenessScore* and *qualityScore* suggested by the evaluators is denoted by c_m and q_m . Considering c_m and q_m as a measure of *completenessScore* and *qualityScore* won't be the correct as their might be malicious evaluators trying to collude or provide other workers with unjust score. So to find and remove such malicious workers that are outliers among the evaluators, we compute the standard deviation c_s of the *completenessScore* and q_s as the standard deviation of *qualityScore* specified by the evaluators. Evaluators whose score is beyond a certain threshold away from the mean in any of the two criteria, we tag him/her as an outlier. Remaining evaluators are said to form consensus. The evaluation of the submission of the evaluators who are found to be outliers are not considered. This helps to deter evaluators from maliciously increasing or decreasing other worker's reputation. If consensus is not reached, evaluators are reassigned and above mentioned steps are repeated. There can be better outlier detection methodology but due the current limitations of Ethereum we stick to this methodology.

Once the consensus is reached, the scores given by the evaluators on two different criteria are combined. Let $completeScore_{jk}$ and $qualityScore_{jk}$ represent the consensus of the evaluator on the completeness and quality, respectively, of the submission k made by worker j . $completeScore_{jk}$ and $qualityScore_{jk}$ can be computed using equation 6.1 and 6.2, respectively.

$$completeScore_{jk} = \frac{\sum_{i=0}^{ec} c_{ik} * r_i}{\sum_{i=0}^{ec} r_i} \quad (6.1)$$

$$qualityScore_{jk} = \frac{\sum_{i=0}^{ec} q_{ik} * r_i}{\sum_{i=0}^{ec} r_i} \quad (6.2)$$

Here ec represents the number of evaluators in consensus and c_{ik} and q_{ik} are the scores for completeness and quality respectively given by the i^{th} evaluator for k^{th} submission. Each of their scores are weighed by the credibility of that evaluator. Credibility is measured in terms of the reputation of the evaluator on the platform and is represented as r_i for i^{th} evaluator.

Hence the final evaluation score $finalScore_{jk}$ assigned to the j^{th} worker for a k^{th} submission is the weighted average of score received in the two criteria. The weight assigned to these criteria can be decided by the task poster and sums up to 1. Suppose w_c is the weight assigned to the $completeScore_{jk}$ and w_q is the weight assigned to the $qualityScore_{jk}$. Then $finalScore_{jk}$ is calculated as :

$$finalScore_{jk} = \frac{w_q * qualityScore_{jk} + w_c * completeScore_{jk}}{w_q + w_c} \quad (6.3)$$

Upon computing $finalScore_{jk}$, it is then sent to the worker as a transaction. By sending a transaction to the worker we are trying to maintain a chain of reviews the worker has ever got on the platform. This makes it easier for anybody to prove the existing reputation of the worker. In this case, no central party can maliciously alter the reputation score, and no worker can fake its reputation on the platform. The score received by the worker is then added to the reputation of the worker.

The reputation is updated only if the worker makes a submission and it is not altered in the case when the worker accepts the task and does no submission.

6.2 Evaluating submission

The other way of increasing reputation is by evaluating the submission made by peer workers. The amount of increase or decrease in reputation of evaluator depends on how far is its evaluation for quality and completeness for a given submission from the $completeScore_{jk}$ and $qualityScore_{jk}$ received by the worker. Let the score received by the i^{th} evaluator upon evaluating submission k

be represented by $eScore_{ik}$. It can be computed using Equation 6.4

$$eScore_{ik} = \frac{100 - |qualityScore_{jk} - q_{ik}| + 100 - |completeScore_{jk} - c_{ik}|}{2} \quad (6.4)$$

6.3 Updating reputation

For every submission the worker does, he/she is required to evaluate y other tasks represented by v . Let α be the weight assigned to the score that the worker receives upon evaluating other submissions to get his/her l^{th} submission evaluated. And let $(1 - \alpha)$ be the weight assigned to the score that the worker receives for his l^{th} submission. α can vary between $[0,1]$. Then the total reputation increased of the worker by doing submission and its corresponding evaluation is given by Equation 6.5. The value of α has been considered to be 0.25 as Alfaro and et al. [17] have shown that taking 25% as value of α provides sufficient incentive for evaluators.

$$repScore_{jk} = (1 - \alpha)finalScore_{jk} + \alpha \frac{\sum_{b=0}^{|v|} eScore_{jb}}{y} \quad (6.5)$$

On the other hand if the worker j is willing to evaluate even though he has no pending submission to be evaluated, then the reputation score increased for that particular submission k 's evaluation is given by Equation 6.6

$$repScore_{jk} = \alpha * eScore_{jk} \quad (6.6)$$

Both the above cases consider only those workers that are not outliers. But when the worker is an outlier, in the latter case the workers are decentralized as their reputation is decreased. The decrease is proportional to their $eScore_{jk}$ and is given by Equation 6.7

$$repScore_{jk} = -\alpha * eScore_{jk} \quad (6.7)$$

For the former case the worker is required to correctly evaluate the tasks before his task can get evaluated. Since evaluators are randomly chosen, the probability of him/her getting the next task early in time is low, hence delaying his submission to be evaluated.

6.4 Reward

The reward received by the worker wholly depends on the $finalScore_{jk}$ he receives for his submission. If task reward as posted by the task porter for task t is $taskReward_t$ then the reward received by the worker j for submission k corresponding to task t represented as $reward_{jk}$ and

can be computed using Equation 6.8

$$reward_{jk} = \frac{finalScore_{jk} * taskReward_t}{100} \quad (6.8)$$

Apart from the reward, the worker also gets back the fee (*acceptanceFees*) that he/she paid while accepting the task. The fees returned (*feeReturned_{jk}*) to the worker depends on the completeness of the submission and is computed using equation 6.9

$$feeReturned_{jk} = \frac{completeScore_{jk} * acceptanceFees}{100} \quad (6.9)$$

The reward and the acceptance fee that remains is sent to the task poster.

Chapter 7

Experiments and Results

In order to perform experiments and test the performance, validity and other parameters of our system, we develop an implementation on Ethereum blockchain using **solidity** which is a programming language used to develop smart contracts. The resulting implementation is organized into 5 different contracts with each smart contract handling a specific group of functions relating to our system, details for which have already been described in Section IV.

We have also developed a front end web interface that makes it easy for the users to interact with our platform and also is able to hide the complexity behind the function calls. We use web3.js, which is a collection of Ethereum JavaScript API libraries that enable our front end to connect to and talk to our smart contract on the Ethereum blockchain. For creating the user interfaces and views we have used Reactjs, a popular JavaScript library for creating user interfaces. We use Metamask for signing transactions. While our system is not tied to any particular storage system, we have used IPFS in our implementation due to its decentralized and distributed nature. Our frontend interacts with IPFS via ipfs-api which is its JavaScript API and using this API allows to add and retrieve files from IPFS.

When a worker registers on our platform, he or she is expected to provide an IPFS link (hash) to his/her profile or portfolio which contains information that would be helpful to task posters while choosing workers for their task. Also while registering, a worker is expected to provide a fee as a deposit which we have chosen in our implementation to be ether equivalent of \$5 (0.0118 ether at the time of writing [9]). It is required of the task poster to send the reward amount to the agreement while creating it. The fees that is to be paid by the worker in order to accept the agreement, is taken to be 25% of the reward for the corresponding task. It is chosen so that a balance is maintained between the compensation that the task poster gets and the. It is chosen so as not to oppress the worker and provide necessary compensation to the task poster.

Once solidity code is compiled, all the functions are converted into low level assembly opcodes suitable for the Ethereum Virtual Machine (EVM). The gas cost or cost of executing each of these low level operations is fixed and defined in the Ethereum yellow paper. So the gas cost for any function is a measure of the cost of its processing requirements and for any function or any

other particular flow of control, the amount of gas required for its successful execution is constant and can be determined deterministically. Wallets such as Metamask also provides an estimate of gas cost for execution of function based on the same idea.

To gain a better idea of the transaction costs involved in executing the major functionalities of our application, we first obtain gas requirements and then estimate their cost by taking a price of 1 Gwei (giga-wei) per gas. It may be noted that 1 billion Gwei is equal to 1 ether and the price of 1 ether at the time of this experiment (June 2018) is \$436.

Table 7.1: Gas costs of executing various functions of our smart contract

Function details	Gas requirement	Cost estimate (in USD)
create worker	229,786	0.100
create task poster	228,410	0.099
post task with fees	250,502	0.109
create agreement	198,134	0.086
accept agreement	49,729	0.021
submit hash	114,068	0.049
assign evaluators	328,702	0.143
first evaluation submit	133,073	0.058
second evaluation submit	105,620	0.046
third evaluation submit	274,360	0.119
Become evaluator	47,878	0.021

Table 7.2: Cost comparison of various crowd sourcing platform

Reward amount(USD)	User type	AMT(USD)	Upwork(USD)	WorkerRep(USD)
50	Worker	Nil	10	0.436
50	Task Poster	10	1.375	0.195
100	Worker	Nil	20	0.436
100	Task Poster	20	2.75	0.195
1000	Worker	Nil	200	0.436
1000	Task Poster	200	27.5	0.195

Using the data from Table 7.1, we calculate the total cost of a task being posted to it being evaluated on our system. The total gas required for this is calculated to be 1,502,066 which translates to \$ 0.631 . The cost of registering as worker or a task poster has not been included in this since those costs are incurred only once. Where as platforms such as Upwork charge 20%, 10%, or 5% from the worker depending on the amount of reward received from the task poster. It also charges a processing fee of 2.75% to the task poster. On the other hand Amazon Mechanical turk charges 20% or more from the task poster of the reward amount, where minimum fees is 0.01\$. In our system the overhead cost other than the reward remains the same from task having less meta data to task having more meta data unlike [30]. Also the overhead cost remains the same for given any amount of reward. The comparison of cost on various platforms can be seen from figure 7.2. During the course of the development of our smart contracts, various options

were explored for testing the smart contracts. One method is to use softwares such as ganache-cli that provide a local version of ethereum blockchain for testing. Another popular option is to use test networks such as Rinkeby or Ropsten. These test networks work in manner identical to the Ethereum main net and only differ in the fact that ether on these networks do not actually have any value and they are only used for testing.

Chapter 8

Analysis of the Design

In this chapter we explain how various objectives of our system are fulfilled.

8.1 Privacy and anonymity

Our system maintains privacy of task submissions by using asymmetric key encryption technique. Apart from the evaluators and task poster, no other users will be able to view the submitted work. Our platform provides pseudo anonymity to its user by not linking an account with its personally identifiable information. Pseudo anonymity is inherent from Blockchain. [31] provides anonymity by not linking transactions from various task performed by the worker. But this would fail to build trust upon each other on the platform. As the actions performed by a user in the past would not be known to the other users making it easier for him/her to mislead other on the platform.

8.2 Decentralization

There is no central authority in-charge of running or maintaining data of the platform. All functions carried out by the users on the platform are validated by the miners on the Ethereum network. There are incentives for user to show a good behavior and disincentive for malicious activities.

8.3 Robustness against various attacks

This section talks about how our design helps prevent or sometimes avoid various attacks as mentioned in 4.1.

- *Unfair rating attack/ Bad mouthing*

- A consensus among various evaluators is being taken. The consensus mechanism involves removing outlier/s. Outlier here means evaluator whose evaluation score is not in line to majority of the evaluators. The final rating does not consider the score given by the outlier/s.
- The reputation of the evaluator is decreased or he/she is given another task for evaluation if he/she is found to be an outlier. This acts as a major deterrent for giving a rating that is not consistent with the work that is done by the peer worker.
- *Reciprocity* : Firstly, the chances of getting selected as an evaluator for that worker are really less, given a number of evaluators and secondly, outliers are being dealt with while computing the final score for the worker, it will be tough for any worker to reciprocate.
- *Ballot Stuffing and collusion Attack* : these are prevented by providing a robust evaluation methodology as described in evaluation in section IV.
- *White washing/ Re entry / Sybil* : This is prevented by adding initial entry fee to the platform. The amount returned to the worker when he/she leaves the platform depends on their reputation at that time .
- *Cold Start Problem* : we have some low paying task on the platform that can be used as a starting point for new workers.

8.4 Case study of worker colluding with evaluators

In this section we analyze the case where the worker colludes with other workers on the platform. Let us consider that j^{th} worker has made submission k corresponding to task t . There are n workers in the pool of potential evaluators, that is the workers who are willing to evaluate and are eligible. c denotes the number of workers colluding with worker i and are part of the pool of potential evaluators.

Potential evaluators are divided into bands based on their reputation. Number of bands depends upon number of evaluators required to evaluate the task, in our case which is equal to 3. Out of the given n potential evaluators, suppose n_1 number of them belong first band, n_2 belong to second and n_3 belong to the third reputation band. One of the worker from each of the reputation band is chosen as an evaluator. Out of c , c_1 number of them belong first band, c_2 belong to second and c_3 belong to the third reputation band.

Assumption that are considered are as follows (1) 3 evaluators are required for evaluating the submissions. (2) $n \gg c$. (3) Non colluding workers form majority in at least more than half of the reputation bands.

A successful collusion attack will only be possible if at least two of the three selected evaluator are colluding with the worker since majority can only then be formed. So the probability of

successful collusion represented as $P(S)$ is :

$$P(S) = 1 - P(\bar{S}) \quad (8.1)$$

$P(\bar{S})$ is the probability of an unsuccessful collusion attack that occurs when either one or none of the evaluators chosen are colluding with the worker. That is

$$P(\bar{S}) = P(\text{zeroColluding}) + P(\text{oneColluding}) \quad (8.2)$$

$$P(\text{zeroColluding}) = \frac{\binom{n_1-c_1}{1} * \binom{n_2-c_2}{1} * \binom{n_3-c_3}{1}}{\binom{n}{3}} \quad (8.3)$$

$$P(\text{oneColluding}) = \frac{\binom{c_1}{1} * \binom{n_2-c_2}{1} * \binom{n_3-c_3}{1} + \binom{n_1-c_1}{1} * \binom{c_2}{1} * \binom{n_3-c_3}{1} + \binom{n_1-c_1}{1} * \binom{n_2-c_2}{1} * \binom{c_3}{1}}{\binom{n}{3}} \quad (8.4)$$

Similarly, probability of two of the selected evaluator represented as $P(\text{twoColluding})$ and probability of all three of the selected evaluator represented as $P(\text{threeColluding})$ can be found out. The probability in Equation 8.2 will always be greater than or equal to the sum of $P(\text{twoColluding})$ and $P(\text{threeColluding})$ as the number of potential evaluators are always greater than c . There are chances that the colluding groups work their way around to a successful collusion in the start due scarcity of potential evaluators in upper reputation bands. These can be handled as initially the platform can have certain evaluators whose job will be to validate the evaluations done by the workers. Once sufficient number of workers have joined the platform then there will be no need for external validations on the evaluations done by worker. This is done so as to secure it from malicious users just like any other blockchain system would initially do.

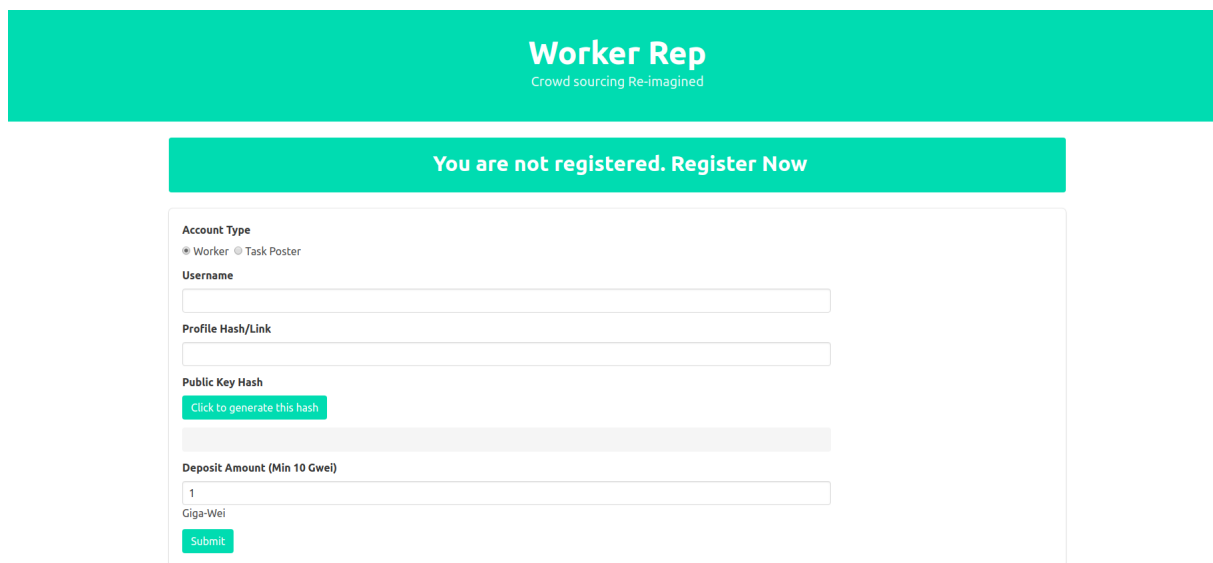
The second and the third assumption above can hold true due to the fact that there is a fee involved for creating users on the platform. Secondly the number of workers in the pool of potential evaluators and their distribution in each reputation band should be such that the choosing colluding evaluator from each of the band should be high and one should be chosen from at least half of the reputation band.

There exist other possibilities such as (1) when a task poster colludes with worker performing the task, the chances of successfully being able to collude is negligible since the submission made by the worker is evaluated by various other workers on the platform. (2) When a task poster colludes with some of the other potential evaluators to decrease the rating of the worker who is assigned to complete the task posted by the task poster. This case is similar to the above case where worker colludes with potential evaluators. (3) The possibility when the task poster and various other workers are colluding can be worked around using the above 2 possibilities.

Chapter 9

WorkerRep:Portal

We also develop a front end web interface that makes it easy for the users to interact with our platform. We use JavaScript, web3.js, Reactjs and Metamask. For storage we have used IPFS.



The image shows a web interface for 'Worker Rep' with a teal header. Below the header is a teal banner that says 'You are not registered. Register Now'. Underneath is a registration form with the following fields and options:

- Account Type:** Radio buttons for 'Worker' (selected) and 'Task Poster'.
- Username:** A text input field.
- Profile Hash/Link:** A text input field.
- Public Key Hash:** A button labeled 'Click to generate this hash' and a corresponding empty text input field.
- Deposit Amount (Min 10 Gwei):** A text input field containing the number '1'.
- Unit:** A label 'Giga-Wei' is positioned below the deposit amount field.
- Submit:** A teal button at the bottom of the form.

Figure 9.1: Blockchain based crowdsourcing platform

Chapter 10

Conclusions, Limitations, Future Work

10.1 Conclusions

In this work, we propose a distributed and secure crowdsourcing platform with a robust reputation management scheme. The existing centralized schemes are susceptible to attacks on central servers or misuse by the central authority. Using the blockchain based technique we build a platform that is less expensive than the existing, does not rely on any third party and overcomes malicious manipulation of reputation and various other attacks on reputation system. It also provides traceability, prevents unvalidated modification of data and gives fair share of reward for the worker and compensation for the task poster. The reputation score provided by the system can be trusted due to immutability inherited by the system from blockchain. And we also provide Proof of concept of our design by building the system on Ethereum test net.

10.2 Limitation

We faced a delay of 2.4 minutes on average [9] while performing experiments between creation of a transaction and its confirmation by other nodes. This delay is expected to reduce with the upcoming transaction scalability updates in Ethereum [8]. Doing computation on Ethereum blockchain is very expensive and increases the cost for the user. Lastly, our design was based on peers evaluating completed tasks and the task poster did not play any role in this step.

10.3 Future work

As a part of future work, we can provide a more effective technique of assigning submissions to worker to evaluate that can relate to the previous tasks that the worker completes and can depend on how recently the worker has completed an evaluation. Computational oracles can be used that reduce cost and perform the computations off-blockchain instead of doing it on blockchain but this is done at the cost of incorporating centralization. Possible modifications

can be explored that allow the task poster to have a reputation score and for him/her to play a greater role in the evaluation step. Performing study to understand the quality of the review and taking measure to ensure quality evaluation.

Bibliography

- [1] Amazon mechanical turk weblink, <https://www.mturk.com/>.
- [2] Meta mask web link, <https://metamask.io/>.
- [3] Topcoder weblink, <https://www.topcoder.com/>.
- [4] Upwork weblink, <https://www.upwork.com/>.
- [5] Iota whitepaper, <https://iota.readme.io/docs/whitepaper>. 1, Oct. 2017.
- [6] J. howe, the rise of crowdsourcing, *wired magazine*, vol. 53, no. 10, pp. 14, Oct. 2006.
- [7] Intuit forecast: 7.6 million people in on-demand economy by 2020, <http://math.tntech.edu/rafal/cliff11/index.html>. Accessed: April 12, 2018.
- [8] Vitalik buterin: Sharding and plasma could scale ethereum by 10,000x, <http://bitcoinist.com/vitalik-buterin-sharding-plasma-scale-ethereum-10000-times/>. Accessed: June 03, 2018.
- [9] Ethereum network information. Accessed: June 28, 2018.
- [10] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.
- [11] Emmanuelle Anceaume, Gilles Guette, Paul Lajoie-Mazenc, Nicolas Prigent, and V Viet Triem Tong. A privacy preserving distributed reputation mechanism. In *Communications (ICC), 2013 IEEE International Conference on*, pages 1951–1956. IEEE, 2013.
- [12] Marcella Atzori. Blockchain technology and decentralized governance: Is the state still necessary? 2015.
- [13] Yukino Baba and Hisashi Kashima. Statistical quality estimation for general crowdsourcing tasks. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–562. ACM, 2013.
- [14] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better—How to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*, pages 399–414. Springer, 2012.

- [15] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.
- [16] Davide Carboni. Feedback based reputation on top of the bitcoin blockchain. *arXiv preprint arXiv:1502.01504*, 2015.
- [17] Luca De Alfaro and Michael Shavlovsky. Crowdgrader: A tool for crowdsourcing the evaluation of homework assignments. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 415–420. ACM, 2014.
- [18] Richard Dennis and Gareth Owen. Rep on the block: A next generation reputation system based on the blockchain. In *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*, pages 131–138. IEEE, 2015.
- [19] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
- [20] Anurag Dwarakanath, NC Shrikanth, Kumar Abhinav, and Alex Kass. Trustworthiness in enterprise crowdsourcing: a taxonomy & evidence from data. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 41–50. ACM, 2016.
- [21] Snehal Kumar Neil S Gaikwad, Durim Morina, Adam Ginzberg, Catherine Mullings, Shirish Goyal, Dilrukshi Gamage, Christopher Diemert, Mathias Burton, Sharon Zhou, Mark Whiting, et al. Boomerang: Rebounding the consequences of reputation feedback on crowdsourcing platforms. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 625–637. ACM, 2016.
- [22] Yang Gao, Yan Chen, and KJ Ray Liu. On cost-effective incentive mechanisms in microtask crowdsourcing. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(1):3–15, 2015.
- [23] Antonio Ghezzi, Donata Gabelloni, Antonella Martini, and Angelo Natalicchio. Crowdsourcing: a review and suggestions for future research. *International Journal of Management Reviews*, 20(2):343–363, 2018.
- [24] John H Hartman, Ian Murdock, and Tammo Spalink. The swarm scalable storage system. In *Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on*, pages 74–81. IEEE, 1999.
- [25] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1, 2009.
- [26] Ori Jacobovitz. Blockchain for identity management, 2016.
- [27] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 2492–2500, 2014.

- [28] Ece Kamar and Eric Horvitz. Incentives for truthful reporting in crowdsourcing. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems-volume 3*, pages 1329–1330. International Foundation for Autonomous Agents and Multiagent Systems, 2012.
- [29] Jörn Klingler and Matthew Lease. Enabling trust in crowd labor relations through identity sharing. *Proceedings of the Association for Information Science and Technology*, 48(1):1–4, 2011.
- [30] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, and Jianan Liu. Crowdabc: A blockchain-based decentralized framework for crowdsourcing. Technical report, IACR Cryptology ePrint Archive 2017: 444, 2017.
- [31] Yuan Lu, Qiang Tang, and Guiling Wang. ZebraLancer: Private and anonymous crowdsourcing system atop open blockchain. *arXiv preprint arXiv:1803.01256*, 2018.
- [32] Ke Mao, Licia Capra, Mark Harman, and Yue Jia. A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software*, 126:57–84, 2017.
- [33] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
- [34] David Oleson, Alexander Sorokin, Greg P Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *Human computation*, 11(11), 2011.
- [35] Ryan Osgood. The future of democracy: Blockchain voting’. *COMP116: Information Security*, 2016.
- [36] Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay’s reputation system. In *The Economics of the Internet and E-commerce*, pages 127–157. Emerald Group Publishing Limited, 2002.
- [37] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. A trustless privacy-preserving reputation system. In *IFIP International Information Security and Privacy Conference*, pages 398–411. Springer, 2016.
- [38] Kyle Soska and Nicolas Christin. Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In *USENIX Security Symposium*, pages 33–48, 2015.
- [39] Mozghan Tavakolifard and Kevin C Almeroth. A taxonomy to express open challenges in trust and reputation systems. *Journal of Communications*, 7(7):538–551, 2012.
- [40] Sin Gee Teo and Amudha Narayanan. Privacy-preserving survey by crowdsourcing with smartphones. 2018.
- [41] Eran Toch. Crowdsourcing privacy preferences in context-aware applications. *Personal and ubiquitous computing*, 18(1):129–141, 2014.

- [42] Sarah Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59(11):15–17, 2016.
- [43] Mark E Whiting, Dilrukshi Gamage, Snehal Kumar S Gaikwad, Aaron Gilbee, Shirish Goyal, Aipta Ballav, Dinesh Majeti, Nalin Chhibber, Angela Richmond-Fuller, Freddie Vargus, et al. Crowd guilds: Worker-led reputation and feedback on crowdsourcing platforms. *arXiv preprint arXiv:1611.01572*, 2016.
- [44] Chunchun Wu, Tie Luo, Fan Wu, and Guihai Chen. An endorsement-based reputation system for trustworthy crowdsourcing. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 89–90. IEEE, 2015.
- [45] Yifan Wu. An e-voting system based on blockchain and ring signature. *Master. University of Birmingham*, 2017.
- [46] Hong Xie, John CS Lui, and Don Towsley. Incentive and reputation mechanisms for online crowdsourcing systems. In *Quality of Service (IWQoS), 2015 IEEE 23rd International Symposium on*, pages 207–212. IEEE, 2015.
- [47] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *Software Architecture (ICSA), 2017 IEEE International Conference on*, pages 243–252. IEEE, 2017.
- [48] Kan Yang, Kuan Zhang, Ju Ren, and Xuemin Shen. Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE communications magazine*, 53(8):75–81, 2015.
- [49] Bin Ye, Yan Wang, and Ling Liu. Crowd trust: A context-aware trust model for worker selection in crowdsourcing environments. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 121–128. IEEE, 2015.
- [50] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms for electronic marketplaces. *Decision support systems*, 29(4):371–388, 2000.
- [51] Omar F Zaidan and Chris Callison-Burch. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1220–1229. Association for Computational Linguistics, 2011.
- [52] Martin Zeilinger. Digital art as “monetised graphics”: enforcing intellectual property on the blockchain. *Philosophy & Technology*, 31(1):15–41, 2018.
- [53] Xiang Zhang, Guoliang Xue, Ruozhou Yu, Dejun Yang, and Jian Tang. Truthful incentive mechanisms for crowdsourcing. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2830–2838. IEEE, 2015.

- [54] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.