Improving the Efficiency of Blockchain Applications with Smart Contract based Cyber-insurance

Jia Xu
Jiangsu Key Laboratory of Big Data Security and Intelligent Processing
Nanjing University of Posts and Telecommunications
Nanjing, China
xujia@njupt.edu.cn

Yongqi Wu
Jiangsu Key Laboratory of Big Data Security and Intelligent Processing
Nanjing University of Posts and Telecommunications
Nanjing, China
Q16010119@njupt.edu.cn

Xiapu Luo
Department of Computing
The Hong Kong Polytechnic University
Hongkong, China
csxluo@comp.polyu.edu.hk

Dejun Yang
Department of Computer Science
Colorado School of Mines
Golden, USA
djyang@mines.edu

***Abstract*—Blockchain based applications benefit from decentralization, data privacy, and anonymity. However, they may suffer from inefficiency due to underlying blockchain. In this paper, we aim to address this limitation while still enjoying the privacy and anonymity. Taking the blockchain based crowdsourcing system as an example, we propose a new smart contract based cyber-insurance framework, which can greatly shorten the delay, and enable the workers to obtain the economic compensation for increased security risk caused by a conflict between the need to provide service quickly and delay in payment. We model the process of determining insurance premium and number of confirmations as a Stackelberg Game and prove the existence of Stackelberg Equilibria, at which the utility of the requester is maximized, and none of the workers can improve its utility by unilaterally deviating from its current strategy. The experimental results show that our framework can definitely improve the time efficiency of crowdsourcing. Particularly, it takes on average only 33% of the time required by the naive blockchain based crowdsouring solution for time-sensitive cases.**

***Index Terms*—blockchain, cyber-insurance, smart contract, game theory**

I. Introduction

The rapid development of blockchain technology in recent years is largely due to the widespread attention of Bitcoin [1]. Based on the blockchain, Bitcoin is a fully point-to-point digital currency, and independent of any centralized nodes. All transaction records are fully documented in Bitcoin. People have discovered that some advantages of blockchain, such as privacy, anonymity. Blockchain can be applied not only in digital currency, but also in other application scenarios such as crowdsourcing [2], energy management [3], supply chain management [4], and resource allocation [5].

Unfortunately, the underlying blockchain technology may also introduce inefficiency to the applications due to the decentralization of the blockchain. The inefficiency is unacceptable to some time-sensitive applications, where the benefits of users might decrease over time rapidly. For example, in medical diagnostic crowdsourcing, the people who provide the medical case do not want to disclose their private medical information. In this case, a decentralized blockchain application system is more convincing to the privacy of the crowd. However, the doctors hope to be able to collect cases similar to patients as soon as possible to help him make the accurate judgments before their condition deteriorates. Many applications such as the crowdsensing for arrival time prediction of buses [23] and real-time monitoring [24] have the real-time requirement.

It is a universal issue to tradeoff the security and time efficiency for time-sensitive blockchain applications. The inefficiency of blockchain applications is largely due to the fact that any node is considered to be potentially untrustworthy based on blockchain assumptions. In order to ensure user security, every operation is considered valid only if it is confirmed by the majority of the nodes. Some studies aim improving the efficiency of blockchain based applications. For example, the studies of [6-8] aim to improve the efficiency of the blockchain by modifying the blockchain protocol. Unfortunately, such modifications may not be quickly adopted by the widely-used blockchain platforms, such as Ethereum [21], and they may also lead to hard fork to the blockchain.

Cyber-insurance is a promising approach to efficiently manage the cyber risks by transferring them to insurers [9]. It can encourage the service providers (the crowd in medical diagnostic crowdsourcing case) to provide their service in a much faster speed since the consumers (the doctors in medical diagnostic crowdsourcing case) of time-sensitive applications are likely to pay insurance premium for the timely service to compensate the potential loss of providers.

Ethereum is able to provide smart contracts, a Turing-complete programming language [12], which enable users to build applications atop open blockchain and also provides a safe and reliable way to implement cyber-insurance [27]. By using smart contracts, we can complete cyber-insurance

This work has been supported in part by the NSFC (No. 61872193), NSF (No. 1717315), and STITP of NJUPT (No. SYB2018019).

without introducing any actual third-party insurance organization, so as not to hurt the advantages of decentralization of blockchain. Since Ethereum is the largest blockchain platform that supports smart contracts, we implement our solution as smart contracts running on Ethereum.

In this paper, we propose the novel framework for the blockchain based crowdsourcing systems. The smart contract based cyber-insurance framework can improve the time efficiency, and enable the workers of crowdsourcing to obtain the premium to avoid the economic loss due to the increased security risk. Our smart contract based cyber-insurance framework can be applied to not only crowdsourcing systems, but also many other blockchain applications through transferring the risk to the consumers of transactions.

The main contributions of this paper are as follows:

- To the best of our knowledge, we are the first to design a smart contract based cyber-insurance framework to improve the time efficiency of blockchain applications.
- Through theoretical analysis, we show that the proposed smart contract based cyber-insurance only takes at most 37.5% and 31.25% of time to perform each crowdsourcing task in Bitcoin and Ethereum, respectively, comparing with naive smart contract.
- We determine the insurance premium and number of confirmations using *Stackelberg Game*. We compute the unique optimal strategy for workers. Further, we show that there exists at least one positive insurance premium that maximizes the utility of requester when the value function of the task decreases sharply with time.
- We implement our framework on the Ethereum private test network using Geth [11]. We show that our framework definitely improves the time efficiency of the crowdsourcing. Particularly, it takes averagely 33% of the time compared with naive blockchain based crowdsourcing in time-sensitive case.

The rest of the paper is organized as follows. Section II reviews the state-of-art research. Section III presents a naive smart contract for blockchain based crowdsourcing. Section IV presents our smart contract based cyber-insurance framework. Section V presents the detailed analysis of our framework. Section VI presents a game theory method to determine the insurance premium and number of confirmations. Performance evaluation is presented in Section VII. We conclude this paper in Section VIII.

II. Related Work

Some previous studies have been devoted to launch crowdsourcing atop open blockchain to achieve decentralization. Zebralancer [2] focuses on privacy and anonymity by combining various technologies, such as zk-SNARKs and smart contracts. CrowdBC [15] is the first one to combine crowdsourcing with blockchain. CrowdBC mainly focuses on the low agency fee by decentralization. Some other research [3] aims at performing crowdsourcing atop blockchain in certain fields. The time efficiency of above mentioned blockchain based crowdsourcing systems relies on the blockchain protocol itself, which may be quite slow.

Some research has been devoted to improving the efficiency of blockchain applications from different perspectives [6-8]. GHOST protocol [6] and full sharding [8] aim at improving the speed of confirming transactions without security loss. A truly distributed ledger system based on a lean graph of cross-verifying transactions is proposed in [7]. These systems need to change the blockchain protocol to improve the confirmation speed of transactions. One of advantages of our method is that there is no need to change the blockchain protocol itself.

A number of cyber-insurance products have been made in the market [10]. There are some studies about blockchain applications using cyber-insurance [16, 17]. [16] discusses the financial issue of introducing smart contract into blockchain, and proposes a framework of insurance against double-spending. But the mechanism proposed in [16] is not applied to any specific application. In [17], risk management is achieved by introducing the role of insurance provider in the blockchain service market. However, none of above work uses cyber-insurance to speed up time-sensitive applications.

Overall, there is no off-the-shelf framework of accelerating most of blockchain applications.

III. Smart Contract for Crowdsourcing

In our blockchain based crowdsourcing system, each user can participate in not only crowdsourcing but also mining blocks. For crowdsourcing, a user can publicize tasks, submit bids, and perform tasks. This means that a user might be a requester of one task and a worker for another. Each user has its unique identity defined by a pair of public and private keys. Since the blockchain technology is Sybil-proof by adopting the PoW (Proof of Work), we do not put restrictions on registration. In other words, a user can arbitrarily pick a pair of keys to hide its true identity. For a pair of keys that is long enough, we can safely assume that the collision is impossible, and users have no incentive to generate multiple pairs of keys.

Without loss of generality, assume that a crowd of workers $U = \{1, 2, \cdots, n\}$ are interested in performing the crowdsourcing task publicized by the task requester in a crowdsourcing activity. We consider the crowdsourcing process as a sealed reverse auction, which is widely used in many crowdsourcing systems [18, 25, 26]. The process of blockchain based crowdsourcing consists of following 5 steps.

(1) Task Announcement: The requester publicizes a task $\Gamma = (PK, Type, SelectionRule(\cdot), PaymentRule(\cdot))$, where PK is the public key of the requester, Type is the type of the task. $SelectionRule(\cdot)$ and $PaymentRule(\cdot)$ are functions of bid profile of workers that output a set of winners S and their payment, respectively. Meanwhile, the requester announces a smart contract for the crowdsourcing activity to the entire network. The smart contract gives the implementation of crowdsourcing operations, including bid, winner selection rule, and payment rule.

(2) Bidding: Once a task is publicized, each worker signs the crowdsourcing contract with the requester, and submits its

bid including the bidding price that the user wants to charge for performing the task.

(3) Winner Selection and Notification: Once there are enough bids, winner set S can be selected by the task requester according to the winner selection rule implemented in crowdsourcing contract. The task requester notifies winners of the selection results.

(4) Answer Collection: The winners perform the task, and submit their answers to the crowdsourcing system. Same as the previous work concerning blockchain based crowdsourcing [2, 3], we also assume that once the task is finished, each worker will get a flag for denoting its work.

(5) Payment: The payment to the workers who have submitted their answers will be determined by the task requester according to the payment rule in crowdsourcing contract. The winners get the payment.

Contract 1 : ***Crowdsourcing Contract***

```
1: function Bidding(Bid)
2:     if the bid is not enough then
3:         record the bid to bid profile B;
4:     end if
5: end function
6: function GetPayment(FinishFlag)
7:     if FinishFlag == true then
8:         payment=PaymentRule(FinishFlag);
9:         send the payment to the winner;
10:    end if
11: end function
12: function PaymentRule(FinishFlag)
13:    ...
14: end function
15: function SelectionRule(B)
16:    ...
17: end function
```

The above crowdsourcing process can be implemented by a smart contract. The naive smart contract puts all crowdsourcing information, including the description of the task, all of bids and answers, on the blockchain, just like the transactions in the original blockchain digital currency. All the nodes in the network can verify the crowdsourcing information authenticity, and decide which block to continue mining on. Satoshi Nakamoto has proved this method to be reliable [1].

The naive smart contract for the blockchain based crowdsourcing system is illustrated in Contract 1. The task requester puts such a smart contract consisting of the winner selection rule, and payment determination rule, on the Ethereum network. All workers can see it, and can call function $Bidding(\cdot)$ to submit their bids. The function $Bidding(\cdot)$ first checks whether there are enough bids. If not, it then adds this bid in bid profile B, which records all bids. Otherwise, the requester runs function $SelectionRule(\cdot)$, which takes bid profile B as input, and returns a set of winners. Each winner will receive a $FinishFlag$ that denotes the result of its work when the task is finished. Then the task requester calculates the payment to the winners according to the function $PaymentRule(\cdot)$. The winners get payment by calling function $GetPayment(\cdot)$.

IV. Smart Contract based Cyber-insurance

The naive crowdsourcing contract can deal with crowdsourcing tasks. However, all transaction information of task, bid, and payment needs to be confirmed by a certain number of following blocks, which greatly reduced time efficiency of crowdsourcing. In Bitcoin, a transaction is confirmed after 6 confirmations, which takes almost one hour. In Ethereum, it also takes about two minutes to confirm a smart contract. For a crowdsourcing system with a lot of time-sensitive tasks, it is unacceptable to wait long time for collecting answers.

In this section, we improve the time efficiency of blockchain based crowdsourcing through reducing the number of confirmations, and propose the smart contract based cyber-insurance to avoid the potential security threats of this reduction for the workers. Let N_0 be the number of confirmations needed for a block to be confirmed by the network. Take the winner selection and notification step as an example, the winners must be confirmed by the following 6 blocks in Bitcoin and 12 blocks in Ethereum, respectively. This means the workers need to wait for a long time before performing tasks. To speed up the confirmation process of crowdsourcing, we reduce the number of confirmations to N_i, $N_i \leq N_0, i \in S$. However, this straightforward method increases the security risk of workers. The confirmed blockchain with length N_i may be replaced by another longer blockchain. If the replacement happened, the workers will not get any payment even they have submitted the answers. Even worse, the potential security threat might be used by some malicious task requesters to conduct the attack against the workers, e.g., double-spend attack [13].

To deal with this problem, we proposed a new method that can transfer the potential risks to the task requesters. We introduce an Ethereum smart contract that serves as a cyber-insurance against the double-spend attack.

Contract 2 : ***Cyber-insurance Contract***

```
1: function CreateInsurance(PK, C)
2:     if enough ether as the requester claims then
3:         store the ether with requester information into
   contract;
4:     end if
5: end function
6: function ClaimCompensation(txPrime)
7:     if txPrime is insured and is attacked then
8:         send stored insurance to the worker;
9:     end if
10: end function
11: function RetrieveInsurance()
12:    if the insured transaction is confirmed then
13:        send ether back to the requester;
14:    end if
15: end function
```

In order to ensure the effect of cyber-insurance, that is, the smart contract of cyber-insurance can protect all the blocks containing the task information from being attacked, both of requester and workers in the crowdsourcing should first sign the cyber-insurance contract illustrated in Contract 2. Once the cyber-insurance contract is signed, the task requester can call function $CreateInsurance\,(\cdot)$ to sign a cyber-insurance with each worker. Let C be the insurance premium. The insurance premium must be included in the call of function $CreateInsurance\,(\cdot)$ that the task publisher sends to the network. If any worker $i \in U$ is attacked by double-spend, he/she can call function $ClaimCompensation\,(\cdot)$ to get the insurance premium of C to compensate the cost of performing task. This function checks if worker i is really attacked and if the cyber-insurance contract is signed by worker i, where $txPrime$ refers to the transaction that is insured. If so, the insurance premium C will be automatically sent to worker i. If N_0 blocks are generated after the cyber-insurance contract is signed, i.e., the submitted answer of worker i is confirmed by the network, the task requester can call function $RetrieveInsurance\,(\cdot)$ to retrieve its insurance premium.

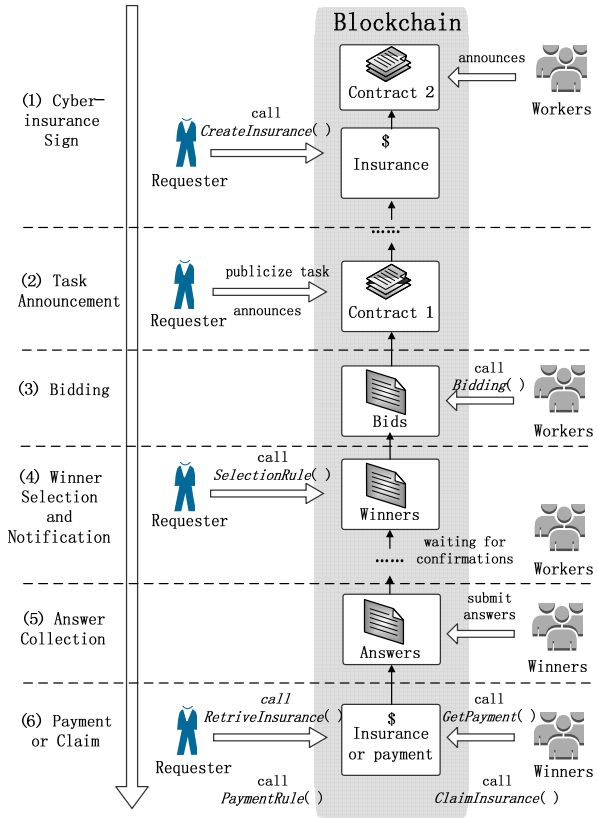

Fig. 1. The improved workflow of blockchain time-sensitive crowdsourcing.

The improved workflow of blockchain based time-sensitive crowdsourcing illustrated in Fig. 1 is as follows:

(1) Cyber-insurance Sign: Each worker announces Contract 2 to the whole network and signs insurance with the requester. For signing Contract 2, the requester determines the insurance premium C, and each worker $i \in U$ determines N_i.

(2) Task Announcement: The requester publicizes a crowdsourcing task and announces Contract 1 to the network.

(3) Bidding: The workers submit the bid and sign Contract 1 with the requester.

(4) Winner Selection and Notification: The requester selects a set of winners S, determines the payment to the winners, and notifies winners of the determination.

(5) Answer Collection: The winners perform the task, and submit their answers to the requester.

(6) Payment or Claim: Each winner waits for the payment. If any worker $i \in U$ is attacked, he/she is paid the insurance premium of C. Otherwise, if there is still no attack after N_0 confirmations, the requester retrieves its insurance premium, and the winners get the payment.

V. Time Efficiency Analysis of Smart Contract based Cyber-insurance

We use T_n and T_c to denote the total time needed to perform a crowdsourcing task using naive crowdsourcing contract and cyber-insurance contract, respectively. Let D be the frequency of block generation. Note that we only aim to speed up the answer collection for the task requesters, and ignore the time for payment confirmation in our time efficiency analysis because the answers have been collected before this step.

First, we analyze the time efficiency of the blockchain based crowdsourcing using naive crowdsourcing contract. Recalling the crowdsourcing process given in Section III, we only consider the time spent in Step (1) to Step (4). Obviously, each of four steps needs time of $N_0 \times D$, and we can conclude straightforwardly that $T_n = 4N_0D$.

Then, we analyze the time efficiency of the blockchain based crowdsourcing using cyber-insurance contract. We use t_1, t_2, t_3, t_4 to represent the time spent from Step (2) to Step (5), in Section IV, respectively. We don't consider the time spent to sign the insurance contract because this can be done before launching the crowdsourcing.

For task announcement step, we can safely believe in the task publicized, and only verify it in winner notification step. When a task is publicized, it should wait for the next block to be included in the blockchain network, thus $t_1 = D$. We apply the same method to the bidding step, and get $t_2 = D$. After all the bids are collected, in Step (4), each winner $i \in S$ needs to wait for N_i confirmations to confirm that both of task and the winner selection are valid before he/she can finally perform the task. Thus, the time spent in this step is $t_3 = D \times \max_{i \in S} N_i$. Finally, we assume that all winners can perform the task fast enough, comparing with the time of block generation, i.e., the next block will contain all the answers of the task. Thus the time spent in this step is $t_4 = D$. Overall, the time spent for each task using cyber-insurance contract is $T_c = t_1 + t_2 + t_3 + t_4 = \left(3 + \max_{i \in S} N_i\right) \times D$.

Theorem 1: The smart contract based cyber-insurance framework takes at most 37.5% and 31.25% of time to

perform each crowdsourcing task theoretically, in Bitcoin and Ethereum, respectively, comparing with naive smart contract.

Proof: Since $\max_{i \in S} N_i \leq N_0$, we have $T_c/T_n \leq (3+N_0)/4N_0$. Recall that N_0 is 6 and 12 in Bitcoin and Ethereum, respectively, we get the theorem. ∎

Since the difficulty of work D relies on the blockchain platform and cannot be changed, T_c totally depends on $\max_{i \in S} N_i$. Note that the above time efficiency analysis can be applied to most blockchain based applications.

VI. Determination of Insurance Premium and Number of Confirmations

In Step (1) of the improved workflow of blockchain based time-sensitive crowdsourcing given in Section IV, each winner $i \in S$ determines N_i, and the requester determines the insurance premium C. We model the determination process as *Stackelberg Game* [19], which we call the Insurance Premium and Confirmation (IPC) game. *Stackelberg Game* is usually used to model the interaction between two types of players: leader and followers. The leader makes its move first. After the leader chooses a strategy, each follower always chooses the best response strategy that maximizes its utility. Knowing this reaction from the followers, the leader chooses a strategy to maximize its utility. The optimal strategies of both leader and followers constitute the *Stackelberg Equilibria*. In IPC game, the requester is leader and the winners are followers. In the first stage, the requester announces its insurance premium C; in the second stage, each winner strategizes its number of confirmations to maximize its own utility.

A. Utilities of Requester and Workers

The utility of the requester is:

$$u_0(N_i) = \sum_{i \in S} v_i(N_i) - \sum_{i \in S} (1 - p(N_i)) r_i - \sum_{i \in S} p(N_i) C - \alpha$$

where $v_i(N_i)$ is the value of the answer provided by worker i. According to the analysis in Section V, we know that the total time of crowdsourcing is linear to the number of confirmations N_i. Since we are discussing the time-sensitive crowdsourcing, $v_i(N_i)$ should be a decrease function of N_i. r_i denotes the payment of crowdsourcing to worker i, which depends on neither the insurance premium nor the number of confirmations. So r_i can be viewed as a constant in this game. $p(N_i)$ is the probability of suffering double-spend attack. To make our model applicable to most cases, we do not specialize $p(N_i)$ here. However, $p(N_i)$ is a convex downward decreasing function of N_i in most existing research [13, 21]. Note that for the requester, the cost of launching double-spend attack increases since we introduce the insurance premium. Thus our smart contract based cyber-insurance enhances the resistance to double-spend attack. Let α be the maintenance fee the requester needs to pay.

The utility of any worker i is:

$$u_i(N_i) = \begin{cases} (1 - p(N_i)) \times r_i + p(N_i) \times C - \beta N_i, & i \notin S \\ 0, & i \in S \end{cases} \quad (1)$$

where β is the maintenance fee per unit time of each worker to perform Contract 2. The maintenance fee is the cost to inform requester periodically that workers are still in work. Since we focus on the time-sensitive tasks, the requester may not be patient enough to wait without periodic message. Moreover, if a malicious worker wants to launch a sybil attack or a flood attack to disturb the crowdsourcing, this maintenance fee gives each worker an extra cost, which is proportional to time, making the cost of launching the attack higher. The total currency depends on the maximum confirmation time (number of confirmations) for all winners. The code structure of Contract 1 and Contract 2 are different, and thus they have different maintenance fees, i.e., $\alpha \neq \beta$.

B. Determine the Number of Confirmations

Given the insurance premium C, the optimal strategy of any winner i can be computed by solving $\max_{0 \leq N_i \leq N_0} u_i(N_i)$. Let N_i^* be the optimal strategy of any winner i. We have:

Lemma 1: Given the insurance premium $C > 0$, the optimal strategy of any winner i is

$$N_i^* = \begin{cases} 0, & C \geq r_i \\ \arg \max_{N_i \in [0, N_0]} u_i(N_i), & r_i > C > r_i - \frac{\beta}{p(0) - p(1)} \\ 0, & 0 < C \leq r_i - \frac{\beta}{p(0) - p(1)} \end{cases}$$

Proof: Based on (1), we rewrite $u_i(N_i)$ as

$$u_i(N_i) = r_i + p(N_i) * (C - r_i) - \beta * N_i$$

Case 1: $C - r_i \geq 0$. In this case, $u_i(N_i)$ is monotonic decreasing with N_i. Let $N_i = 0$, we have $u_i(0) = r_i + p(0) * (C - r_i) > 0$. Thus $N_i^* = 0$ in this case.

Case 2: $C - r_i < 0$. We observe the value of $u_i(N_i + 1) - u_i(N_i)$ for $\forall N_i \in [0, N_0 - 1]$:

$$u_i(N_i + 1) - u_i(N_i) = (p(N_i + 1) - p(N_i)) * (C - r_i) - \beta \quad (2)$$

Recall that $p(N_i)$ is a convex downward decreasing function of N_i, we have $1 > p(0) > p(1) > \cdots > p(N_0) > 0$ and

$$p(0) - p(1) > p(1) - p(2) > \cdots > p(N_0 - 1)\, p(N_0) > 0 \quad (3)$$

We can see that the value of $u_i(N_i + 1) - u_i(N_i)$ decreases with $N_i \in [0, N_0 - 1]$ based on (3). This means $u_i(N_i)$ is a concave function. We further suppose that $p(N_0 - 1) - p(N_0)$ is very close to 0 since it provides adequate resistance to double-spend attack. Let $N_i = N_0 - 1$, and substitute it into (2), we can conclude that $u_i(N_0) - u_i(N_0 - 1) < 0$.

To analyze the monotonicity of $u_i(N_i)$, we consider the value of $u_i(1) - u_i(0)$ in the following two cases:

Case 2.1:

$u_i(1) - u_i(0) = (p(1) - p(0)) * (C - r_i) - \beta \geq 0$, i.e., $C \leq r_i - \frac{\beta}{p(0) - p(1)}$. Note that $u_i(N_i)$ is a concave function and $u_i(N_0) - u_i(N_0 - 1) < 0$, there must be an $N_i^* = \arg \max_{N_i \in [0, N_0]} u_i(N_i)$.

Case 2.2: $u_i(1) - u_i(0) < 0$, i.e., $C < r_i - \frac{\beta}{p(0)-p(1)}$. $u_i(N_i)$ is a monotone decreasing function on $N_i \in [0, N]$. In this case, $u_i(0) = (1 - p(0)) r_i + p(0) > 0$. Thus $N_i^* = 0$. ∎

C. Requester Utility Maximization

According to the above analysis, the requester, which is the leader in the *Stackelberg Game*, knows that there exists a unique optimal strategy N_i^* for each winer i for any given value of C. Hence the optimal strategy of the requester can be computed by solving $\max u_0(N_i^*), i \in S$. We have:

Lemma 2: Given the optimal strategy profile of workers, there exists the optimal strategy $C^* > 0$ of requester if

$$\sum_{i \in S} \frac{dv_i(N_i^*)}{dc}_{C=0} > \sum_{i \in S} p(N_i^*) - \sum_{i \in S} \frac{dp(N_i^*)}{dc}_{C=0} r_i$$

Proof: The derivative of $u_0(N_i^*)$ when $C = 0$ is

$$\frac{du_0(N_i^*)}{dc}_{C=0} = \sum_{i \in S} \frac{dv_i(N_i^*)}{dc}_{C=0} + \sum_{i \in S} \frac{dp(N_i^*)}{dc}_{C=0} r_i - \sum_{i \in S} p(N_i^*).$$

To get a positive insurance premium, we need to let $\frac{du_0(N_i^*)}{dc}_{C=0} > 0$, i.e.,

$$\sum_{i \in S} \frac{dv_i(N_i^*)}{dc}_{C=0} > \sum_{i \in S} p(N_i^*) - \sum_{i \in S} \frac{dp(N_i^*)}{dc}_{C=0} r_i \quad (4)$$

There must exist at least one optimal strategy $C^* > 0$ that maximizes $u_0(N_i^*)$ ∎

Lemma 1 and Lemma 2 lead to the following theorem:

Theorem 2: The *Stackelberg Equilibria* of IPC game with positive insurance premium exists if

$$\sum_{i \in S} \frac{dv_i(N_i^*)}{dc}_{C=0} > \sum_{i \in S} p(N_i^*) - \sum_{i \in S} \frac{dp(N_i^*)}{dc}_{C=0} r_i$$

Note that (4) describes the time-sensitivity of crowdsourcing tasks since $v_i(N_i)$ is the value function of tasks with time. This means that the positive insurance premium exists only if the value decreases sharply with the time.

VII. Performance Evaluation

We implemented the smart contract based cyber-insurance framework on Ethereum private test network composed of one requester and 100 workers. Geth v1.8.27 [11] is used for mining blocks and signing smart contracts. In addition, an extra miner with a 2.6GHz Intel Core i7 CPU is responsible for maintaining the system. To verify the robustness of our crowdsourcing system, we set three different value functions to represent time-sensitive task, slight time-sensitive task, and time-insensitive task, respectively: $v_i(N_i) = \frac{2}{N_i+1}$, $v_i(N_i) = \log_2\left(4 - \frac{N_i}{10000}\right)$, and $v_i(N_i) = 2$. For $p(N_i)$, we use results from [13], and set $p(N_i) = 1 - \sum_{m=0}^{N_i} C_{N_i+m-1}^{m}(1-\lambda)^{N_i}\lambda^m - (1-\lambda)^m\lambda^{N_i}$ when $\lambda = 10\%$. α and β are set to 1e-3 ether for maintaining the network. The average time of block generation is one second. For crowdsourcing, the bids are generated from Cartier 3-day auctions data set [20], which is normalized such that the highest bidding price is one ether. We select the first 10 workers to perform the crowdsourcing task, and apply VCG payment rule [12]. Each measurement is averaged over 1000 instances.

A. Strategies of Requester and Workers

Fig. 2 shows the insurance premium with different maintenance fee of each worker. We can see that the insurance premium decreases slightly with increasing maintenance fee for the time-sensitive task and slight time-sensitive task. We know that the utility of any worker will decrease when the maintenance fee increases according to (1). So the workers will have little incentive to delay their service, and the requester can give fewer premiums to convince workers. Moreover, when the task is time-insensitive, the insurance premium is small since there is not much stimulation for the requester to give the insurance premium.

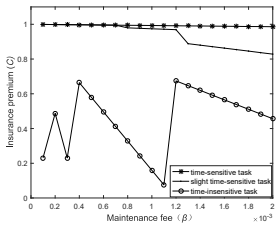

Fig. 2. Strategy of the requester

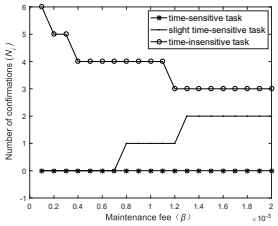

Fig. 3. Strategy of the workers

From Fig. 3, we can see that, regardless of the value of β, the requester will always convince workers to accelerate the crowdsourcing service for all kinds of tasks. It is an amazing observation that the workers give different strategies for three kinds of tasks. For the time-insensitive task, as the analysis given above, the insurance premium is small, and the workers have to cut down the confirmations in order to reduce the cost. For the slight time-sensitive task, the insurance premium can cover the cost when the maintenance fee is low. However, when the maintenance fee increases, the workers tend to gain the payment from the requester because the insurance premium is not sufficient to cover the maintenance fee. Since the requester provides the highest insurance premium to the workers for time-sensitive task, the workers always submit their answers as long as the winner selection is finished, even when the maintenance fee is high.

B. Utilities of the Requester and Workers

The utility of requester or worker depend on the results of IPC game. Fig. 4 and Fig. 5 depict the utilities of the requester and workers, respectively. For the time-sensitive task and slight time-sensitive task, we already know that the requester tends to reduce the insurance premium when the maintenance fee increases according to Fig. 2. Thus the requester's utility will increase accordingly. On the other side, as maintenance fee increases, the average utility of the worker decreases since they

need to pay more maintenance fee. Note that the workers in time-sensitive task will gain more utility than other tasks. This is because the requester gives highest insurance premium for the time-sensitive task. Accordingly, the requester's utility in time-sensitive task is lower than those in slight time-sensitive task and time-insensitive task.

C. Running Time

We measure the running time of crowdsourcing using smart contract based cyber-insurance under different N_i. The value of N_i is in fact determined by the IPC game. Here we set N_i to certain values in order to observe the impact of N_i on running time. We can see from Fig. 6 that the running time is linear increased to N_i. This consistent with our time efficiency analysis of smart contract based cyber-insurance.

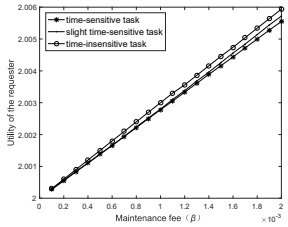

Fig. 4. Utility of the requester

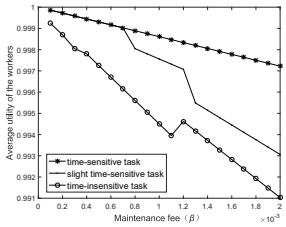

Fig. 5. Average utility of the workers

We can see from Fig. 7 that the running time increases when the number of tasks goes up. Our smart contract based cyber-insurance framework can improve the time efficiency for all three kinds of tasks. Comparing with the naive smart contract (Contract 1), it takes only averagely 33% running time in time-sensitive case. For the cases of slight time-sensitive task and time-insensitive task, the running time is about 45% and 78% of the naive one, respectively.

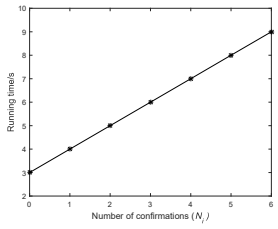

Fig. 6. Running time with different number of confirmations

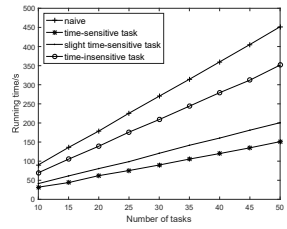

Fig. 7. Running time with different number of tasks

VIII. Conclusion

In this paper, we have designed the smart contract based cyber-insurance framework to accelerate the blockchain based crowdsourcing system. The proposed framework can improve the time efficiency, and enable the workers of crowdsourcing to obtain the economic compensation for increased security risk. We have determined the insurance premium and the number of confirmations using *Stackelberg Game*, and shown that there exists unique *Stackelberg Equilibria*. The experimental results show that our smart contract based cyber-insurance framework can definitely improve the time efficiency of crowdsourcing.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Manubot*, 2009.

[2] Y. Lu, Q. Tang, G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," in *Proc. IEEE ICDCS*, pp. 853-865, 2018.

[3] S. Wang, A. Taha, J. Wang, "Blockchain-Assisted Crowdsourced Energy Systems," in *Proc. IEEE PESGM*, pp. 1-5, 2018.

[4] K. Toyoda, et al., "A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain," *IEEE Access*, vol.5, pp. 17465-17477, 2017.

[5] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource Allocation and Consensus on Edge Blockchain in Pervasive Edge Computing Environments," in *Proc. IEEE ICDCS*, pp.1476-1486. 2019.

[6] A. Kiayias, G. Panagiotakos, On Trees, "Chains and Fast Transactions in the Blockchain," in *Proc. International Conference on Cryptology and Information Security in Latin America*, pp. 327-351, 2017.

[7] X. Boyen, C. Carr, T. Haines, "Blockchain-free cryptocurrencies: A framework for truly decentralised fast transactions," *Cryptology ePrint Archive*, Report, no. 871, 2016.

[8] M. Zamani, M. Movahedi, M. Raykova, "RapidChain: A Fast Blockchain Protocol via Full Sharding," *IACR Cryptology ePrint Archive*, pp. 460, 2018.

[9] R. Pal, L. Golubchik, K. Psounis, et al., "On a way to improve cyber-insurer profits when a security vendor becomes the cyberinsurer," in *Proc. IEEE IFIP Networking* , pp. 1-9, 2013.

[10] A. Marotta, F. Martinelli, S. Nanni, et al., "Cyber-insurance survey," *Computer Science Review*, vol. 24, pp. 35-61, 2017.

[11] Go Ethereum, https://geth.ethereum.org/downloads/, [Online].

[12] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8C37, 1961.

[13] M. Rosenfeld, "Analysis of Hashrate-Based Double Spending," *arXiv preprint*, no. 1402.2009, 2014.

[14] N. Atzei, M. Bartoletti, T. Cimoli, "A survey of attacks on ethereum smart contracts," in *Proc. International Conference on Principles of Security and Trust*, pp. 164-186, 2017.

[15] M. Li, J. Weng, A. Yang, et al., "CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing," *IEEE Trans. Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251-1266, 2018.

[16] Y. Velner, J. Teutsch, L. Luu, "Smart contracts make bitcoin mining pools vulnerable," in *Proc. International Conference on Financial Cryptography and Data Security*, pp.298-316, 2017.

[17] S. Feng, W. Wang, Z. Xiong, et al., "On Cyber Risk Management of Blockchain Networks: A Game Theoretic Approach," *arXiv preprint*, no.1804.10412 , 2018.

[18] J. Xu, J. Fu, D. Yang, et al., "FIMI: A constant frugal incentive mechanism for time window coverage in mobile crowdsensing," *Journal of Computer Science and Technology*, vol. 32, no. 5, pp. 919-935, 2017.

[19] D. Yang, G. Xue, J. Zhang, et al., "Coping with a smart jammer in wireless networks: A Stackelberg game approach," *IEEE Trans. Wireless Communications*, vol. 12, no. 8, pp. 4038-4047, 2013.

[20] modelingonlineauctions, http://www.modelingonlineauctions.com/datasets, [Online].

[21] C. Grunspan, R. Prezmarco, "Double spend races," *arXiv preprint* no. 1702.02867, 2017.

[22] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1-32, 2014.

[23] P. Zhou, Y. Zheng, and M. Li, "How Long to Wait? Predicting Bus Arrival Time With Mobile Phone Based Participatory Sensing," *IEEE Trans. Mobile Computing*, vol. 13, no. 6, pp. 1228-1241, 2014.

[24] J. Xu, J. Xiang, D. Yang, "Incentive Mechanisms for Time Window Dependent Tasks in Mobile Crowdsensing," *IEEE Trans. Wireless Communications*, vol. 14, no. 11, pp. 6353-6364, 2015.

[25] J. Xu, Z. Rao, L. Xu, D. Yang, T. Li, "Incentive Mechanism for Multiple Cooperative Tasks with Compatible Users in Mobile Crowd Sensing via Online Communities," *IEEE Trans. Mobile Computing*, DOI: 10.1109/TMC.2019.2911512.

[26] J. Xu, J. Xiang, Y. Li. "Incentivize maximum continuous time interval coverage under budget constraint in mobile crowd sensing," *Wireless Networks*, vol. 23, no. 5, pp. 1549-1562, 2017.

[27] T. Chen , Y. Zhu , Z. Li, et al., "Understanding Ethereum via Graph Analysis," in *Proc. IEEE INFOCOM*, pp.1484-1492, 2018.