Reliable Decentralized Oracle with Mechanisms for Verification and Disputation

1st Limao Ma, 2nd Kosuke Kaneko, 3rd Subodh Sharma, 4th Kouichi Sakurai
1stDepartment of Informatics Graduate School of Informtion, Science and Electrical Engineering, Kyushu University, Japan
Email:malimao_66@yahoo.co.jp
2ndCybersecurity Center, Kyushu University, Japan
Email:kaneko.kosuke.437@m.kyushu-u.ac.jp
3rdIndian Institute of Technology, Deli, India
Email:svs@iitd.ac.in
4thDepartment of Informatics Faculty of Information, Science and Electrical Engineering, Kyushu University, Japan
Email:sakurai@inf.kyushu-u.ac.jp

***Abstract*—Smart contract using Blockchain technology provides a mechanism to automatically exchange "cash" and "service" according to programmed conditions without requiring reliable third-party intervention. This results in reduction of time and cost for complex contract execution. Some contract execution require external information outside Blockchain as a trigger to execute the code specifying process for a certain contract. However, because Blockchain technology itself does not provide a function to directly access such external information, these applications require a proxy system called "oracle". Oracle is in charge accessing external information, to verify it, and to write it on Blockchain. To avoid security incidents such as oracle writing malicious information on Blockchain, reliability of oracle must be required. This paper introduces a decentralized oracle equipping with verification and disputation mechanisms. To evaluate reliability of the proposed mechanisms, a simulation-based experiment was conducted. The experimental results showed that our solution could effectively suppress the interference of malicious participants and obtained reliable consensus results even if relatively many malicious participants joined in the consensus process on the proposed decentralized oracle.**

***Index Terms*—Blockchain, smart contract, decentralized oracle, system security**

I. Introduction

A. Blockchain

Blockchain [1] network is a P2P network with high Byzantine fault tolerance, and every user in this network shares the same ledger data in their own system. Transaction data is stored as a list of time ordered encrypted blocks. Each block in Blockchain contains a cryptographic hash of previous block, a nonce for Proof-of-Work, and its own transaction data. Because of consensus mechanism, once the contents stored in a block by this construction are modified, data in subsequent blocks must also be tampered. Bitcoin [2] is one of the representatives of Blockchain applications. With the introduction of Turing-complete smart contract [3], Blockchain platforms such as Ethereum [4] are developed. As a result, decentralized applications can run above the Blockchain network, expanding its application field. Since most Blockchain-based applications require external data to facilitate the execution of smart contracts, external data feedback for smart contracts becomes an important issue.

B. Smart Contract

Smart contract [6] is a special computer protocol designed to digitally facilitate, validate or enforce contract negotiation and performance. Compared with traditional contract, smart contract enables trusted transactions even without trusted third parties by a trusted mechanism. The mechanism provides better security and makes commission fee lower. In Bitcoin network, users can create and customize simple smart contracts through a Turing incomplete scripting language. In another way, smart contracts with Turing-complete [7] programming languages are built into certain Blockchains, such as Ethereum, making it possible to create customized smart contracts with more complex logic on the Blockchain. Ethereum users record and write the execution results of smart contract into blocks, keeping program execution history and results consistent throughout the network.

C. Oracle

Decentralized applications based on Blockchain development can be categorized into two categories according to the data sources required for their internal smart contracts: intra-chain data triggering, and extra-chain data triggering [8]. The former is similar to the currency system such as Bitcoin. Its circulation of money within the Blockchain network can be carried out without the need of extra-chain information. The other category needs to face a problem with smart contract: oracle issues.
Since the execution of smart contract is passively triggered, smart contract needs to receive a certain data in order to execute the previously written contract terms. Contrary to the deterministic and consistent environment in Blockchain, outside world is uncertain, any data from outside cannot be written directly in Blockchain. If a smart contract trigger condition is extra-chain data information, the information needs to be written into Blockchain in another way. The

mechanism for writing extra-chain data information into Blockchain is often referred as oracle mechanism [5]. Oracle's main function is to provide external data for Blockchain, enforcing the relevant treaties in smart contract. External data is primarily information related to the conditions which trigger smart contract terms. They can be temperature at a certain time, stock price, match results, and so on. Oracle collects and provides requested data in a secure, reliable way so that Blockchain can interact with the outside world, which means oracle is the interface between Blockchain and real world.

D. Contribution

A secure link between smart contracts in Blockchain and outside world is critical, while oracle provides an important function for communicating data inside and outside Blockchain. Existing solutions each have their own advantages and can be utilized to maximize their application in certain specific situations, which means further research on general solution is still an issue. In this paper, the reliability of proposed mechanism was evaluated by a simulation-based experiment. Base on the effectiveness of experiment result, our proposal will provide a hint for the developers exploring oracle solutions with better generality and security.

II. Related Methods

A. Oracle Architecture Types

Architecture of oracle can be divided into two types; Centralized oracle or Decentralized oracle. Centralized type is a comparatively practical architecture to achieve the role of oracle than Decentralized type because of its simple architecture. The architecture locates one oracle in a smart contract application. The oracle reads information which are in outside of Blockchain and writes it on a code of smart contract. Centralized oracle is indicated several problems such as a single point of failure (SPOF), non-transparent processing that users cannot verified their data on the chain [19].On the other hand, Decentralized oracle does not cause SPOF because of the decentralized architecture. However, Decentralize oracle also has problems. For example, the input of unstructured data requires manual input, which means more time is needed to collect as much input as possible in order to obtain reliable data.

B. Centralized Oracle

Provable, also known as Oraclize, [9] is an Amazon Web Service based oracle service designed to provide data feedback for smart contracts and Blockchain applications. The concept of Oraclize focuses on proving that the data obtained from original data source is genuine and untampered. Oraclize itself does not interfere with the choice and reliability of data source. Although it is a centralized project and costly, it still has a large user base because of its practicality. Town Crier [10] is a centralized oracle that provides data feedback based on Ethereum. It focuses on providing proven data feedback for smart contract and Blockchain applications via Inter Software Guard Extension. However, Town Crier has been pointed out the types of APIs and data feedback that can be provided are limited. It is also referred to be susceptible to SPOF for its centralization.

C. Decentralized Oracle

Chainlink [11] was originally a decentralized oracle network on the Ethereum platform. Its main purpose is to provide reliable data tamper-proof input and output for smart contracts by accessing key data resources through designated APIs. Chainlink requests and provides data through incentives and aggregation models. However, it is pointed out that there are problems such as large chain aggregation cost and low scalability. ASTRAEA [12] is also a decentralized oracle that determines the credibility of voter submission data primarily by using different roles (voter and verifier) to conduct voting games. By splitting roles and incentives of users, voters' choices are influenced by randomness, it can resist malicious voting attacks to a certain extent. Kleros [16] is a decentralized contract dispute arbitration application based on Ethereum. It relies on game theory incentives to ensure that referees properly rule cases.
Augur [13], like Gnosis [14], is a Blockchain-based decentralized forecasting market. It predicts future real-world events through crowd intelligence as oracle's output, and writes the predictions into Blockchain. Augur was strongly influenced by Truthcoin [15] and suppressed the fraudulent behavior of participants by introducing report and dispute systems. At the same time, due to the limitations of its own prediction mechanism, there will be a big challenge to introduce Augur into other applications except forecasting market.

III. Our Proposal Oracle

In this paper, we present a decentralized oracle approach introducing the concept of "reporters" and "verifiers". Also the approach provides a more selective and secure oracle mechanism through a reputation system. In our proposal, requesters can filter the participants for their requests by setting reputation level limit. The higher the level limit requester set, the more reliable data feedback requester get. Participants with higher reputation will be more likely to be reporters, whose role is to collect data requested by smart contracts and report their answers to oracle. On the other hand, participants who cannot become reporters will join as verifiers. Their role is to collect requested data, and submit their answers with an additional amount of reputation to oracle.

A. Roles of participants

1) Requester: A requester generates a request (*Request* explained as formula 1) through oracle's smart contract. When oracle accepts the request, it will be added to the task list which contained all requests from the whole oracle network.

Participants can obtain all requests from the task list through synchronization information of oracle nodes.

2) Reporter: When a reporter participates in a request, as a means of suppressing Sybil attack, a certain of reputation will be deposited as a proof of participation. The reporter must submit its answer (*Reporter1Report* explained as formula 2) to oracle's smart contract within a given time. In this request, if the submitted answer is consistent with final consensus result of the request, the reporter can receive a reward (cryptocurrency) and the deposited reputation will be returned. Otherwise, the reporter cannot receive any reward. As a penalty, the deposited reputation will be paid as a bonus for other participants.

3) Verifier: A verifier also needs to deposit a certain of reputation as a proof of participation. The verifier is required to submit its answer (*Verifier1Report* explained as formula 3) to oracle and stack an additional reputation for its own claim within a given period of time. Reputation weighted calculation will be processed for verification result based on all reports submitted by verifiers. The result with the highest reputation will be chosen as consensus result to verify the reports from reporters and verifiers. The reports consistent with the consensus result will become "correct". Those verifiers will be considered as "correct" participants and receive rewards according to their respective contributions. In the proposed method, only the "correct" verifiers can recover their deposited reputation and obtain a certain percentage of reputation as reward.

B. Whole Process

The overall process of our proposal is described in Fig1 and Fig 2.

1) Request Phase: **StepI**: A user submits a request to oracle based on its smart contract as a requester (1-1 in Fig.1). Before the requester submit its request, its ability to meet payment terms will be checked and its property will be confirmed.

StepII: Oracle's smart contract generates a request based on user's requirements and adds it to the task list of oracle network(1-2 and 1-3in Fig.1). Each request contains at least the following information. Each element in the information is described such as "element: type".

Request = { 'Task id' : string, 'Task content' : string, 'Reporter number' : integer, 'Reporter4reputation' : float, 'Verifier4reputation' : float, 'Fee2reporters' : float, 'Fee2verifiers' : float }(formula 1)

'Reporter number' refers to the maximum number of participants who can participate in this request as a reporter. *'Reporter4reputation'* refers to the reputation that a reporter needs to deposit beforehand as a proof of participating in this request. *'Verifier4reputation'* is the reputation that a verifier needs to deposit. In general, the reporter needs to deposit more reputation than the verifier. *'Fee2reporters'* and *'Fee2Verifier'* refer to the rewards of "correct" reporters and verifier, respectively.

2) Report Phase: **StepIII**: After the oracle participants find the request, and if the number of participants does not exceed the specified number (*'Reporter number'*), they can deposit the required reputation (*'Report4reputation'*) in advance as a proof of participation to join as reporters(2-1 and 2-2 in Fig.1). Reporters need to collect requested data and submit them to oracle within the deadline of certain time period (2-3 and 2-4 in Fig.1). A report submitted by a reporter contains at least the following information. Each element in the information is described such as "element: type".

Reporter1Report = { 'Task id' : string, 'User id' : string, 'Answer' : string, 'Report reputation' : float, 'Statue' : 'Reported', 'Time' : float }(formula 2)

3) Verification Phase: **StepIV**: If the number of participants exceeds *'Reporter number'*, subsequent participants can only join as verifiers (3-1 in Fig.1). The verifiers deposited a portion of reputation (*'Verifier4reputation'*) as a proof of participation. They are required to submits their own answers within the deadline of certain time period(3-2 and 3-3 in Fig.1). Unlike reporters, verifiers need to stack an extra reputation (*'Stack'*) on their own answers (3-4 in Fig.1). A report submitted by a verifier contains at least the following information. Each element in the information is described such as "element: type".

Verifier1Report = { 'Task id' : string, 'User id' : string, 'Answer' : string, 'Stack' : float, 'Verifier reputation' : float, 'Statue' : 'Verified', 'Time' : float }(formula 3)

StepV: After the verification phase over, reputation weighted calculation will be performed. In detail, among the reports submitted by all verifiers, the one with the highest reputation will become consensus result (3-5 in Fig.1).

4) Dispute Phase: **StepVI**: After a consensus result appears by the end of verification phrase, if several people indicate that they do not agree with the result, they can join in the request for the consensus result they want to dispute (4-1 in Fig.2). This part of participants will become disputers for this request. Similar to ordinary verifiers, they need to deposit a portion of the reputation (*'Verifier4reputation'*) and stack additional reputation on a new answer to raise a dispute phrase (4-2 and 4-3 in in Fig.2). Only when the reputation stacked on the new answer exceeds two times the old consensus result's, the disputers can form a new consensus result. After that, a new round of dispute is observed (4-4 in Fig.2). When disputers failed to form their answer or there is no more dispute on the consensus result by deadline, reward distribution will be occurred (5-1 in Fig.2).

5) Reward Phase: **StepVII**: In reward phase, every report submitted by participants will be compared with the consensus result. The participants whose reports are consistent with the consensus result will be considered as "correct" participants (include reporters, verifiers and disputers) (5-2-1 in Fig.2), the other will be "incorrect" participants (5-2-2 in Fig.2). All "incorrect" participants will not be able to receive any

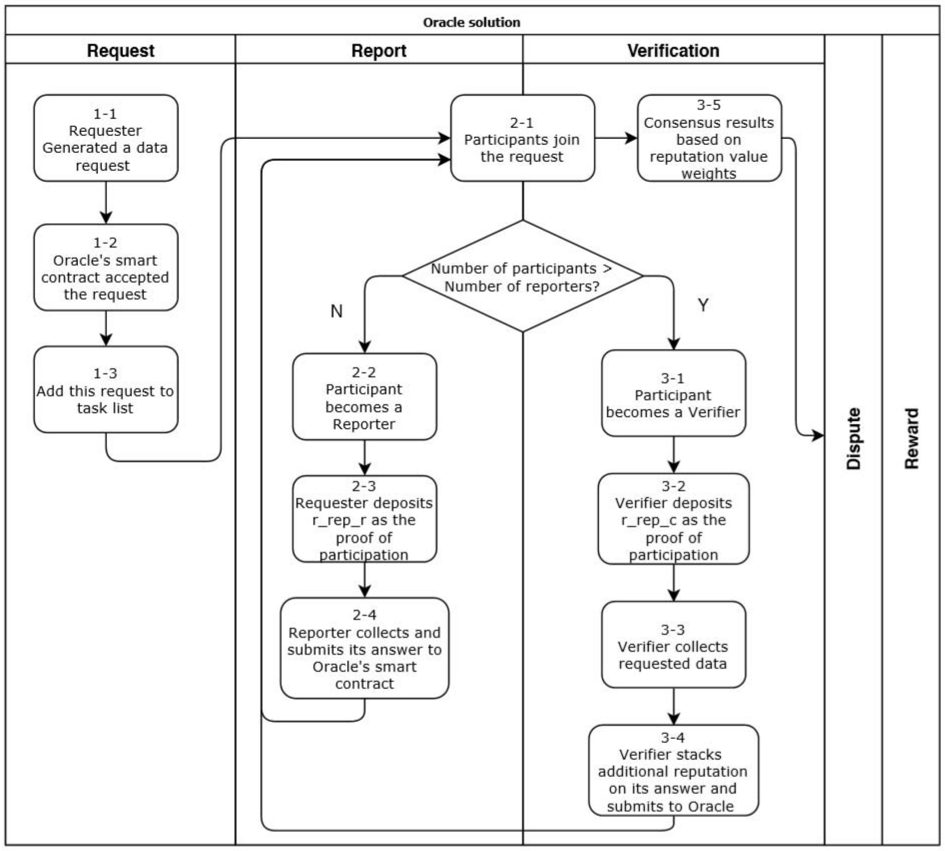
Oracle solution
Request
Report
Verification
Dispute
Reward
1-1 Requester Generated a data request
1-2 Oracle's smart contract accepted the request
1-3 Add this request to task list
2-1 Participants join the request
3-5 Consensus results based on reputation value weights
Number of participants > Number of reporters?
N
Y
2-2 Participant becomes a Reporter
2-3 Requester deposits r_rep_r as the proof of participation
2-4 Reporter collects and submits its answer to Oracle's smart contract
3-1 Participant becomes a Verifier
3-2 Verifier deposits r_rep_c as the proof of participation
3-3 Verifier collects requested data
3-4 Verifier stacks additional reputation on its answer and submits to Oracle

Fig. 1. Overall process of the solution(part 1)

reward set by the requester, but they lose all of the reputation they have deposited (5-3-2 in Fig.2). The "correct" reporters will average the reward *'Fee2reporters'* (cryptocurrency). The "correct" verifiers and disputers will receive an additional reputation as rewards (5-3-1 in Fig.2). For example, we assume that ***n*** reporters participated in ***request0***, among them ***n1*** reporters are considered as "correct" participants. The sum of verifiers and disputers are ***m***, and ***m1*** of them are considered as "correct" participants. By the time of reward phase, there is a total of ***Trp*** reputation stacked on the final consensus result, ***Frp*** reputation stacked on others. For reporter ***R1*** who is considered as "correct" participant in ***request0***, the reward ***R1rwd*** for ***R1*** is as follow(1):

$$R1rwd = \frac{feer}{n1} \tag{1}$$

Among them, ***feer*** is the encrypted currency reward (*'Fee2reporters'*) set by requester. For verifier ***V1*** who stacked ***Trpa*** reputation on its own answer and be considered as "correct" participant, the reputation reward ***V1grp*** for ***V1*** is as follow(2):

$$\begin{aligned} V1grp = & \frac{Trpa}{Trp} \times [feec + (n - n1) \times rrepr + \\ & (m - m1) \times rrepc + Frp + (rrepc + Frpa) \end{aligned} \tag{2}$$

Specifically, ***feec*** represents the reputation reward (*'Fee2verifiers'*) set by requester. And ***rrepr*** refers the proof of participation (*'Reporter4reputation'*) deposited by reporters. (***rrepc + Frpa***) is the refund deposit for ***V1***, including the proof of participation (*'Verifier4reputation'*) and the stack on ***V1***'s answer.

IV. Simulation Evaluation

We evaluated our proposal method by conducting experiments through several simulations. In this simulation, we created 100 participants including reporters, verifiers and disputers. All of them will obtain benefits from a feasible range. The reputation (***rp***) initially owned by each participant is set to 1000 ***rp***. To join in a request, reporters need to deposit 100 ***rp***, verifier and disputers need to deposit 10 ***rp***. The request for simulation was set as follow:

Request = { 'Task id' : 0x001, 'Task content' : T or F, 'Reporter number' : 10, 'Reporter4reputation' : 100.00,

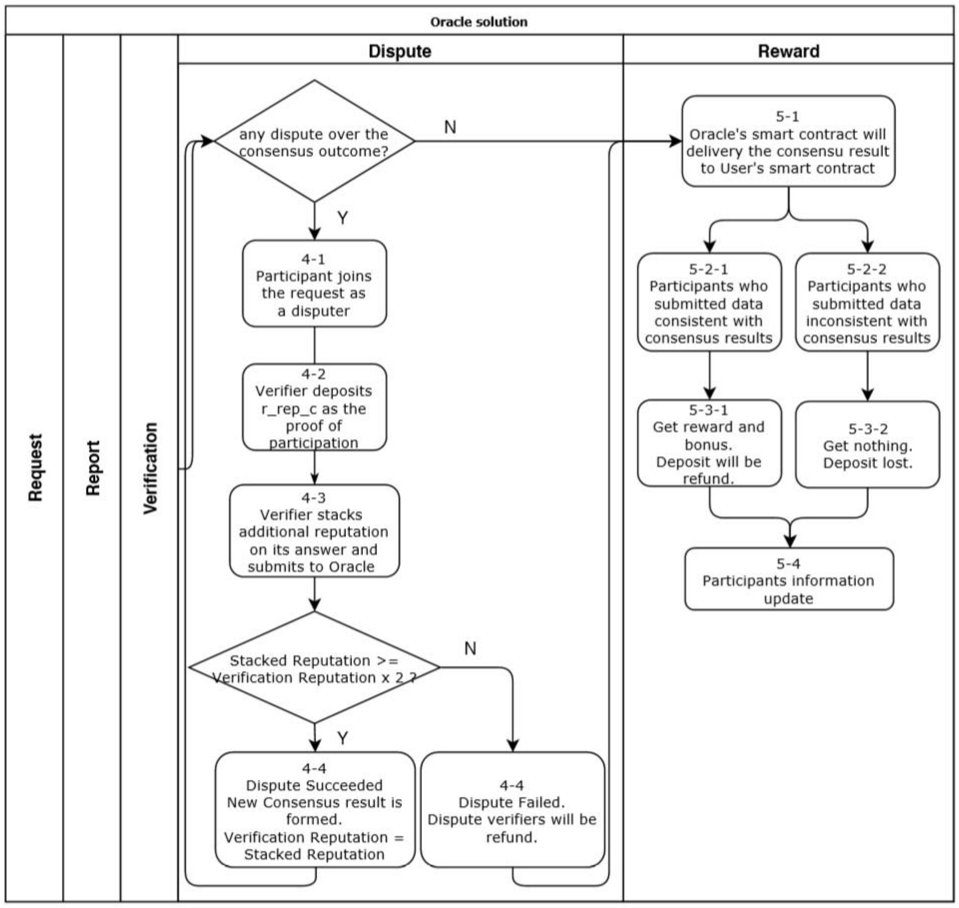
Oracle solution
Request
Report
Verification
Dispute
Reward
any dispute over the consensus outcome?
N
Y
4-1 Participant joins the request as a disputer
4-2 Verifier deposits r_rep_c as the proof of participation
4-3 Verifier stacks additional reputation on its answer and submits to Oracle
Stacked Reputation >= Verification Reputation x 2 ?
4-4 Dispute Succeeded New Consensus result is formed. Verification Reputation = Stacked Reputation
4-4 Dispute Failed. Dispute verifiers will be refund.
5-1 Oracle's smart contract will delivery the consensu result to User's smart contract
5-2-1 Participants who submitted data consistent with consensus results
5-2-2 Participants who submitted data inconsistent with consensus results
5-3-1 Get reward and bonus. Deposit will be refund.
5-3-2 Get nothing. Deposit lost.
5-4 Participants information update

Fig. 2. Overall process of the solution(part 2)

'Verifier4reputation' : 10.00, 'Fee2reporters' : 10.00, 'Fee2verifiers' : 1000.00 }

The maximum possible of verifiers' number was defined as ***PPL***. The number of verifiers for each simulation was randomly chosen from [1,***PPL***]. In addition, we set the maximum for the reputation that verifiers can stack on their answers (***max=3×rrepr***) in order to avoid the abuse of reputation. We assume that there are ***P*** honest and ***Q*** dishonest participants in every request. When a dishonest participants became reporter, the reporter will have 100% to submit a wrong answer (**F**). A dishonest verifier submits a correct answer (**T**) by proportion of ***P1***, while a dishonest disputer submits the same one by proportion of ***P2***. For a certain ***Q***, we will release a same request for 10000 times. When reward phase was ended, the percentage of the final consensus result which formed by the honest participants will be calculated. To evaluate our proposal method, we conducted a comparative experiment. As comparative method, we prepared a majority voting-based consensus method called m-first voting in research field of Volunteer Computing. Also, to evaluate the effect of dispute phase, we compared two cases of proposal method with dispute phase and without dispute phase. We evaluated the performance of each method in 4 cases shown below. The results are shown in Fig.3 - Fig.6.
CASE1:PPL=30,P1=50%,P2=50%.
CASE2:PPL=50,P1=50%,P2=50%.
CASE3:PPL=30,P1=30%,P2=30%.
CASE4:PPL=30,P1=0%,P2=0%.

The horizontal axis represents the proportion of current dishonest participants (***Q***), while the vertical axis represents the accuracy of the consensus results at this ratio. At the same time, we observed the impact of the proportion of dishonest participants on the frequency of disputes.
It can be seen from the above data that the correct rate of the three consensus schemes shows different degrees of decline with the dishonest increases without considering the Nash equilibrium. However, the proposal maintains a fairly high rate of accuracy when the proportion of dishonest participants reaches 50%, especially around 30%. If we considering that the dishonest participants have motivation to submit correct answer, the performance of the proposal is better than the

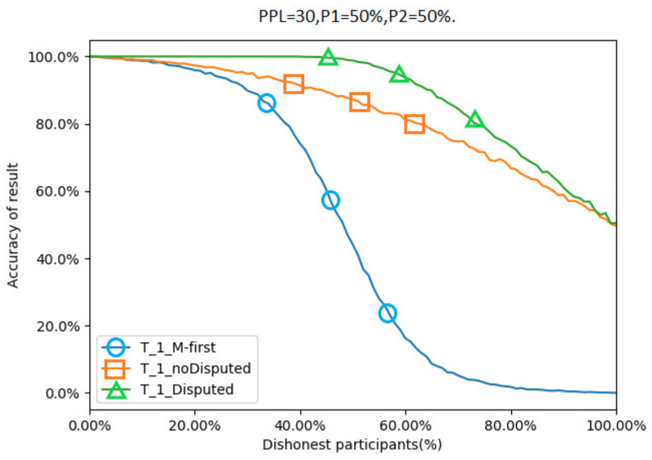

Fig. 3. Comparative result in CASE1

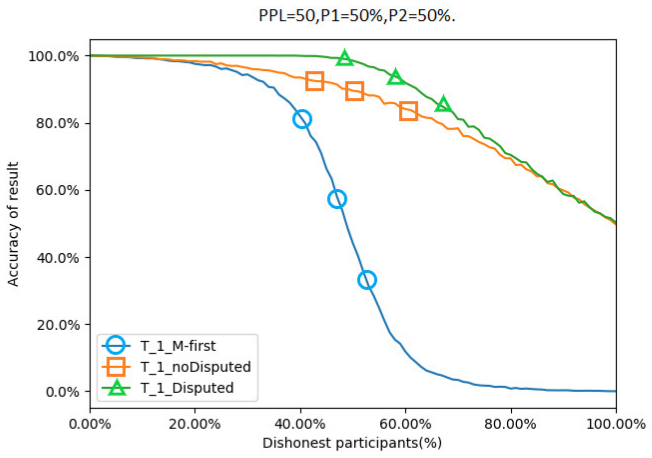

Fig. 4. Comparative result in CASE2

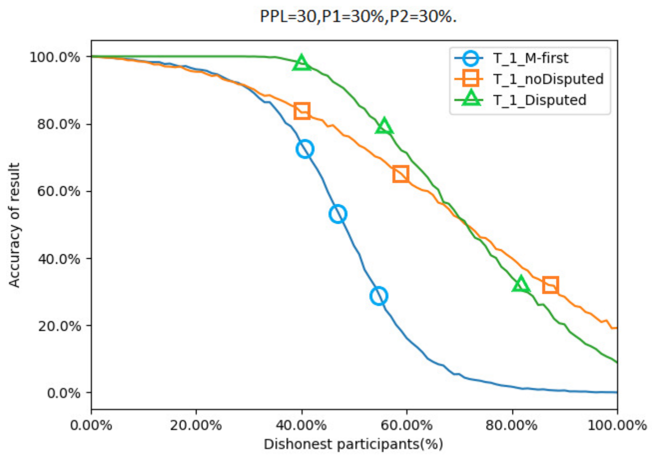

Fig. 5. Comparative result in CASE3

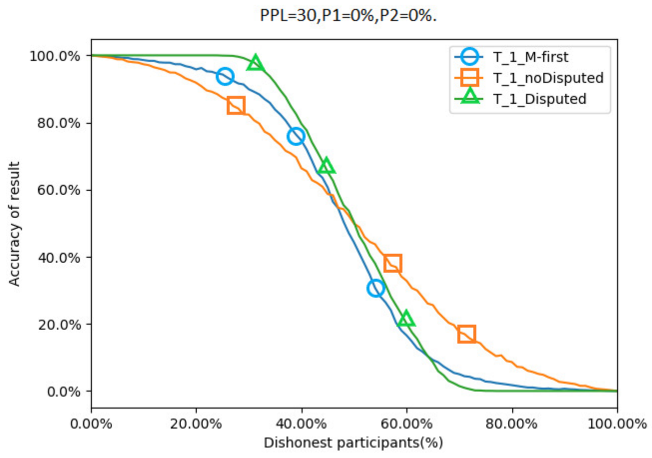

Fig. 6. Comparative result in CASE4

other two comparative method to varying degrees.
In [18], a similar incentive mechanism without dispute phase was proposed. A dishonest verifier with the highest reputation can bribe reporters and easily controll consensus results with a relatively small cost. It is because there is no mechanism of reconsideration for other honest verifiers with relatively low reputation to overturn the result they want to dispute. For example, when the answer of a request is whether Ture or False, malicious reporters and verifiers only need to submit False to the system and form two "False" consensus results while correct answer is True. After the consensus are established, since the results are consistent with each other, the malicious will not only be exiled from the system, but also obtain considerable benefits, thus declining the reliability of system. Based on this, we consider the introduction of a dispute mechanism can suppress this phenomenon to some extent. When dishonest verifiers try to form an incorrect consensus result with a small amount of cost, the remaining participants can dispute it and form a new consensus through the dispute phase. Once the new consensus is established, the dishonest need more manpower and reputation to overturn it. As an incentive, honest participants can earn the reputation of the dishonest verifiers in from the dispute if it failed to overturn the new consensus result, thus gaining a high reputation level. A higher reputation level means that it is easier to become a reporter and obtain cryptocurrency from other requests.

V. Security Analysis

A. Reliability

1) Malicious Reporter: Since requesters can set the lowest reputation (*'Reporter4reputation'* and *'Verifier4reputation'*) as a proof of participation, it is necessary for participants to gain a higher level of reputation so that they are able to become participants in most requests as reporters. When a malicious reporter submitted incorrect data to oracle, due to the high deposit for the proof of participation, it has a high risk of losing the deposit, as well as the reward from request it joined. On the other hand, requesters can increase the minimum reputation requirement for their requests. As a result, participants with low reputation will be blocked by this, thereby suppressing the possibility of malicious attacks.

2) Malicious Verifier: Since requesters can filter out verifiers with low reputation by raising minimum reputation of proof of participation, it is assured for malicious verifier to obtain the reputation in advance. Furthermore, verifiers need to stack enough reputation on their own answers, or they will probably failed to form their consensus result in verification phase. A feasible way to controll the verification phase is to work with other malicious verifiers. However, success rate of this method is quite low because of the exsistence of dispute phase. Even if the verifiers succeeded to controll the verification phase, honest participants can overturn the verification result formed by malicious verifiers. Moreover, to overturn a previous consensus result requires twice the reputation stacked on it.
In the proposal, restrictions on reputation and dispute mechanisms will make Sybil Attacks almost impossible. It is because that no one can become a participant in any requests with no reputation. Verifiers' reward is reputation, which means they will not gain actual benefits (cryptocurrency) from requests. However, they will increase their reputation level, being more competitive in the verification and dispute phase of other requests. Owing to the fact that getting real profits requires a high level of reputation, one of the incentives to be a verifier is to earn reputation, ultimately become a reporter for most requests to get real benefits (cryptocurrency).

B. Bribery tolerance

If malicious participants want to benefit from the proposed mechanisms, they need to work together to controll consensus results. Truthcoin introduced a kind of concept called "double incentives", enabling participants to vote anonymously. It claimed that voters have incentives to confuse other voters, reminding them that they are lying, so that they can share the reward with the least voters. In this way, malicious participants are unable to accurately controll the number of people who actually voted for wrong answers. In our proposal, participants are anonymous to each other, and malicious participants have an incentive to betray others. It demonstrates that there is also such incentive mentioned above.
The P + epsilon attack [17] is characterized by the fact that attackers pay their bribes only if their attacks failed. In other words, the attackers are able to attack systems without actually paying any price if the attack succeeded. In this paper, since the reputation of "incorrect" participants will be paid to "correct" verifiers and disputers, when the benefits that participants can obtain from being honest are more valuable than bribes, there is high possibility of betraying attackers.

VI. Conclusion

In this paper, we present a reliable decentralized oracle approach. The proposal performs role differentiation and consensus result calculation based on participants' reputation. A simulation-based experiment was conducted to evaluate the reliability of the proposal. It showed that the interference of malicious participants was suppressed, and reliable consensus results could be obtain even if relatively many malicious participants joined in the consensus process of the proposal. As future work, we will try to program smart contracts for the proposal and deploy it on Ethereum for further research.

Acknowledgment

This research was supported by the National Institute of Science and Technology Corporation (JST) Strategic International Collaborative Research Program (SICORP) and the grant of JSPS Grant-in-Aid for Scientific Research JP18H03240.

References

[1] Brito, J., Castillo, A. (2013). Bitcoin: A primer for policymakers. Mercatus Center at George Mason University.
[2] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.
[3] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. white paper, 3, 37.
[4] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.
[5] Xu Zhong, Zou Chuanwei. (2018). What can the blockchain do and can't do? Financial research, 461(11), 1-16. http://www.jryj.org.cn/CN/Y2018/V461/I11/1
[6] Bashir, I. (2018). Mastering blockchain: Distributed ledger technology, decentralization, and smart contracts explained. Packt Publishing Ltd.
[7] Governatori, G., Idelberger, F., Milosevic, Z., Riveret, R., Sartor, G., Xu, X. (2018). On legal contracts, imperative and declarative smart contracts, and blockchain systems. Artificial Intelligence and Law, 26(4), 377-409.
[8] Oraclechain Technical White Paper. (2017) http://oraclechain.io/files/oraclechain_white_paper_en.pdf
[9] Provable Documentation. (2019). https://docs.provable.xyz/
[10] Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E. (2016, October). Town crier: An authenticated data feed for smart contracts. In Proceedings of the 2016 aCM sIGSAC conference on computer and communications security (pp. 270-282). ACM
[11] Ellis, S., Juels, A., Nazarov, S. (2017). Chainlink: A decentralized oracle network. Retrieved March, 11, 2018.
[12] Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N., Kastania, A. (2018, July). Astraea: A decentralized blockchain oracle. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 1145-1152). IEEE.
[13] Peterson, J., Krug, J. (2015). Augur: a decentralized, open-source platform for prediction markets. arXiv preprint arXiv:1501.01042.
[14] Gnosis Whitepaper(2017) https://gnosis.io/pdf/gnosis-whitepaper.pdf
[15] Sztorc, P. (2015). Truthcoin, peer-to-peer oracle system and prediction marketplace.
[16] Clément Lesaege, Federico Ast. (2018).Kleros Short Paper v1.0.6 kleros.io/assets/whitepaper.pdf
[17] Vitalik Buterin. (2015). The P + epsilon Attack https://blog.ethereum.org/2015/01/28/p-epsilon-attack/
[18] Shota Johjima, Kosuke Kaneko, Subodh Sharma, Kouichi Sakurai.(2019). "Simulation of Secure Volunteer Computing by Using Blockchain", The Proceedings of the 33rd International Conference on Advanced Information Networking and Applications (AINA-2019) , pp.883-894, Kunibiki Messe, Matsue, Japan Mar. 27-29, 2019.
[19] Orisi White Paper(2014) https://github.com/orisi/wiki/wiki/Orisi-White-Paper