An Examination Protocol for Handling Programmable Answers Using a Public Blockchain

Shu Takayama
Tokyo Institute of Technology
Tokyo, Japan
takayama.s.ae@m.titech.ac.jp

Yuto Takei
Tokyo Institute of Technology
Tokyo, Japan
takei.y.aj@m.titech.ac.jp

Kazuyuki Shudo
Tokyo Institute of Technology
Tokyo, Japan
shudo@is.titech.ac.jp

***Abstract*—Examinations that involve programming, such as programming contests and certification exams, are common. However, current examination protocols require full trust in the operating organizations, which allows malicious organizations to cheat. As a countermeasure, efforts are being made to develop examination protocols that do not need a trusted authority.**

One way to build such a decentralized protocol is to apply blockchain. Several blockchain-based examination protocols have been proposed, but they can only handle a finite set of correct answers.

We propose an examination protocol that can handle a set of correct answers defined by a program and mathematically guarantee the validity of the protocol. We also implement a programming contest platform based on the proposed method on Ethereum and show the feasibility of the method. Furthermore, we measure the economic cost of the method using the implementation and confirm that the method is feasible for approximately 30–40 USD.

***Index Terms*—blockchain, examination, Ethereum, competitive programming**

I. Introduction

For the healthy operation of society, various types of examinations, such as proficiency tests and entrance exams, need to be conducted without fraud. However, current examination protocols are based on full trust in the operating organizations, which has led to incidents of malicious organizations falsifying results [1].

The more fraud is uncovered, the more trust operating organizations need to gain. This trend will promote oligopolies in the examination market and encourage individual exams to become larger. This makes auditing more difficult and fraud easier. One way to overcome this negative feedback loop is to build examination protocols that do not need a trusted authority.

Soon after Nakamoto [2] proposed Bitcoin, an electronic currency without a trusted central authority, people realized that the core technology of Bitcoin, the blockchain, could be used for applications other than currency. For example, Namecoin [3] provides a decentralized version of name registration and resolution services traditionally provided by a DNS. Such decentralized applications have come to be known as *dapps*. With this background, it was natural to propose an examination protocol using blockchain.

The pioneering work of Yoshimura [4], Sawada [5], and Kaneko et al. [6] has convinced us of the possibility of building blockchain-based examination protocols. However, the methods of these previous studies have the problem that they can handle only a finite set of correct answers and cannot be used for exams that require complex correct and incorrect judgments, such as programming contests.

Based on these previous studies, we propose a decentralized examination protocol that can handle a correct answer set defined by a program. The proposed method is described using an original model of computation, and the validity of the method is mathematically guaranteed. We also implement a decentralized programming contest platform based on the proposed method on Ethereum [7]. In addition, we propose solutions to problems that emerged during the implementation process and measure the economic cost of the method using the implementation.

This paper is organized as follows: In Section 2, we clarify the problems of previous studies and specify the properties that the proposed method should satisfy. In Section 3, we introduce a computational model to describe the proposed method. In Section 4, the basic *proposed method* α is discussed. In Section 5, we discuss the *proposed method* β, with reduced economic cost. In Section 6, we propose solutions to problems that emerged in the process of implementing the proposed method and measure the economic cost of the method using the implementation.

II. Previous Studies

Yoshimura [4] was the first to propose an examination protocol using dapp. Although Yoshimura's method focuses on a form of security contest called capture-the-flag (CTF), it can be extended to a general-purpose examination protocol that handles a finite set of correct answers with some trivial modifications. Some problems with Yoshimura's method are that it requires a fixed set of answerers before an exam, and only the fastest correct answerer in each exam is recorded. The economic cost is $O(|P||A|)$ for a questioner and $O(1)$ for an answerer, where P is a set of answerers and A is a set of correct answers[1].

This work was supported by JSPS KAKENHI Grant Number JP21H04872. This work was supported by SECOM Science and Technology Foundation.

[1]When implemented in a Bitcoin pay-to-script-hash (P2SH) transaction, $O(1)$ is the cost for a questioner, and $O(|P||A|)$ is that for an answerer.

Sawada [5] combined Yoshimura's method with a commitment scheme and improved it to handle an unlimited number of answerers. The economic cost of Sawada's method is $O(|A|)$ for both a questioner and an answerer, where A is a set of correct answers.

Kaneko et al. [6] proposed an examination protocol using dapp independently of Yoshimura and Sawada. Unlike Yoshimura's and Sawada's methods, Kaneko et al.'s method records all correct answerers. However, as in Yoshimura's method, it is necessary to fix a set of answerers before an exam. The economic cost is $O(|P|)$ for a questioner and $O(|A|)$ for an answerer, where P is a set of answerers and A is a set of correct answers.

Some common problems for these existing methods are that they can handle only a finite set of correct answers and that they do not discuss how to distribute the content of a problem.

Based on the problems with these existing methods, we propose a method that satisfies the following four points: First, it can handle an unlimited number of answerers. Second, all correct answerers are recorded. Third, it can handle a set of correct answers defined by a program. Fourth, the integrity of the problem content can be guaranteed.

III. Model of Computation

Let $k \in \mathbb{N}$ be a security parameter, and let $N \in \mathbb{N}$ be sufficiently large. Let $\|$ be the concatenation operator for two sequences of bits, and let λ be the empty sequence.

A. Cryptographic Hash Function

We define a cryptographic hash function as a 5-tuple $\left(\{HK_k\}_{k\in\mathbb{N}}, \mathrm{Gen}, l, \mathrm{Hash}, CO\right)$ satisfying the following seven conditions:

- (Hash key set) For each $k \in \mathbb{N}$, HK_k is a nonempty set of binary sequences.
- (Hash key generation function) Gen is a probabilistic polynomial-time algorithm. For each $k \in \mathbb{N}$, if 1^k is input, it outputs $hk \in HK_k$.
- (Hash length) $l : \mathbb{N} \to \mathbb{N}$
- (Hash function) Hash is a polynomial-time algorithm. For each $k \in \mathbb{N}$, if $hk \in HK_k$ and $x \in \{0,1\}^*$ are input, it outputs $y \in \{0,1\}^{l(k)}$.
- (Compatible oracle set) CO is a nonempty set of oracles.
- (Collision resistance) Let $\mathcal{A}$ be any probabilistic polynomial-time algorithm that is arbitrarily interrogatable to each oracle belonging to CO', a finite subset of CO. $\mathbb{P}(\mathrm{Coll}_{k,\mathcal{A}} = 1)$ is a negligible function for k, where $\mathrm{Coll}_{k,\mathcal{A}}$ is a probabilistic game defined by the following:
  1) $hk \leftarrow \mathrm{Gen}(1^k)$.
  2) $(x, x') \leftarrow \mathcal{A}(hk)$.
  3) If $x \neq x' \wedge \mathrm{Hash}(hk, x) = \mathrm{Hash}(hk, x')$, return 1; otherwise, return 0.
- (Preimage resistance) Let $\mathcal{A}$ be any probabilistic polynomial-time algorithm that is arbitrarily interrogatable to each oracle belonging to CO', a finite subset of CO. $\mathbb{P}(\mathrm{Pre}_{k,\mathcal{A}} = 1)$ is a negligible function for k, where $\mathrm{Pre}_{k,\mathcal{A}}$ is a probabilistic game defined by the following:
  1) $hk \leftarrow \mathrm{Gen}(1^k)$.
  2) Choose x uniformly from $\bigcup_{n=0}^{N} \{0,1\}^n$.
  3) $x' \leftarrow \mathcal{A}(hk, \mathrm{Hash}(hk, x))$.
  4) If $\mathrm{Hash}(hk, x) = \mathrm{Hash}(hk, x')$, return 1; otherwise, return 0.

In what follows, we assume that there is one cryptographic hash function.

B. Parties

We model a system as a sequence of probabilistic polynomial-time algorithms. Each algorithm has a unique *address* of n_a bits. Algorithms send and receive messages with each other. In other words, a party is a pair of an address and a probabilistic polynomial-time algorithm. We denote the address of a party $\mathcal{P}$ as $a_{\mathcal{P}}$.

We consider two types of parties: *external parties*, which represent system participants outside a blockchain, and *contract parties*, which represent programs on a blockchain.

External parties issue a send command to send a message. Messages are expressed in a list format, where the first term of each message is the destination address. Each sent message is pushed to the *message queue* of the destination, with the first term replaced by the source address. To prevent contract parties from sending messages to each other and creating a loop, contract parties do not send messages.

Parties issue a receive command to pop a message from their message queue. If a message queue is empty, it returns a list with all terms set to λ. In addition, as described below, there are timing and frequency restrictions on receiving commands for contract parties.

C. Execution Model

Execution of a system proceeds in phases.

In the first phase of each execution, an *initialization function* is first executed to determine the initial arguments of each party. The arguments of an initialization function are given as the arguments of an entire system.

At the beginning of each phase, external parties are in an activated state, and contract parties are in a deactivated state. Contract parties are activated when they receive a message from another party and issue a receive instruction only once immediately after activation. This restriction replicates the behavior of dapps on Ethereum. On the other hand, external parties are never activated by receiving a message. If an external party is deactivated during a phase, it remains deactivated until the start of the next phase.

Each party runs for a polynomial time with one activation. This time restriction also applies to a party that deviates from a protocol due to a failure. When a party is deactivated and then reactivated, the return value of the previous action is used as the argument for the next action. When all the parties are deactivated, the next phase is started.

D. Openness of Contract Parties

The initial arguments of contract parties, the return values of all actions of contract parties, and all messages addressed to

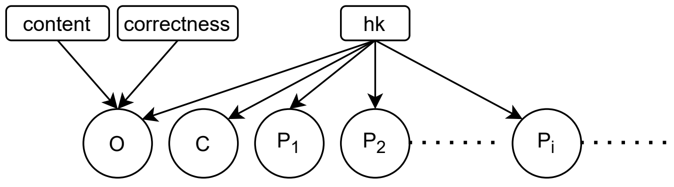

Figure 1. The parties in the proposed method α.

contract parties are recorded and can be accessed by external parties. This is because contract parties run on a blockchain.

E. Failure Model

Contract parties do not fail, because they run on a blockchain. External parties can fail, but the polynomial-time restriction still applies.

IV. Proposed Method A

The proposed method α is designed as a dapp on a blockchain. We describe the protocol using the computation model introduced in Section 3.

A. Parties

The hash key is the same for all parties. Figure 1 illustrates the parties in this method.

- Questioner $\mathcal{O}$ is an external party that holds an exam. $\mathcal{O}$ has the hash key $hk \in \bigcup_{k\in\mathbb{N}} HK_k$, the exam content $content \in \bigcup_{n=0}^{N} \{0,1\}^n$, and the correctness judgment program $correctness \in \bigcup_{n=0}^{N} \{0,1\}^n$ as the initial arguments[2].
- Contract $\mathcal{C}$ is the only contract party required by our method. C has the hash key $hk \in \bigcup_{k\in\mathbb{N}} HK_k$ as its initial argument.
- Answerer $\mathcal{P}_i$ is an external party that takes an exam. $\mathcal{P}_i$ has the hash key $hk \in \bigcup_{k\in\mathbb{N}} HK_k$ as their initial argument. They can also interrogate an oracle $answer_i \in CO$, which outputs an answer of at most $N - n_a$ bits when the exam content is input.

B. Intuitive Description

1) (Announcement start phase) $\mathcal{O}$ sends $\mathrm{Hash}(hk, content)$ and $\mathrm{Hash}(hk, correctness)$ to $\mathcal{C}$. $\mathcal{C}$ records the received data.
2) (Announcement phase) $\mathcal{O}$ announces an exam and invites answerers. The channel used for the announcement is not specified in this method.
3) (Submission start phase) When it is time to start the exam, $\mathcal{O}$ sends $content$ to $\mathcal{C}$. $\mathcal{C}$ computes the cryptographic hash of the received data and verifies that it matches the one recorded in Step 1. If it matches, the received data are recorded as $content'$.
4) (Submission phase) Each $\mathcal{P}_i$ sends $\mathrm{Hash}(hk, answer_i(content')\|a_{\mathcal{P}_i})$ to $\mathcal{C}$ to submit their answer[3].
5) (Judgment start phase) When it is time to end the exam, $\mathcal{O}$ sends the correctness to $\mathcal{C}$. $\mathcal{C}$ computes the cryptographic hash of the received data and verifies that it matches the one recorded in Step 1. If it matches, the received data are recorded as $correctness'$.
6) (Judgment phase) Each $\mathcal{P}_i$ sends $answer_i(content')$ to $\mathcal{C}$. $\mathcal{C}$ computes the cryptographic hash of the bitstring concatenation of the received data and the source address and verifies that the hash matches the hash recorded in Step 4. If it matches, the received data are judged by $correctness'$, and the result of the judgment is recorded.

[2]The judgment program must be a pure function. In other words, the program must not change its behavior according to the environment surrounding it. Otherwise, the results of some judgments will be unable to be uniquely determined.

Algorithm 1 Announcement start phase

```
1: function INITIALIZE($hk$,$content$,$correctness$)
2:     Let the initial arguments of $\mathcal{O}$ be ($hk$,$content$,$correctness$).
3:     Let the initial arguments of $\mathcal{C}$ be ($hk$,$\lambda$,$\lambda$,$\lambda$,$\lambda$,$\{\lambda\}$,$\lambda$,$\{\lambda\}$).
4:     Let the initial arguments of $\mathcal{P}_i$ be ($hk$).
5: end function
6: procedure $\mathcal{O}$($hk$,$content$,$correctness$)
7:     send $(a_{\mathcal{C}}, \mathrm{Hash}(hk, content), \mathrm{Hash}(hk, correctness))$
8:     return ($hk$,$content$,$correctness$)
9: end procedure
10: procedure $\mathcal{C}$($hk$, $phase$, $hContent$, $hCorrectness$, $content'$, $\{hAnswer_a\}$, $correctness'$, $\{correct_a\}$)
11:     $(from, d_1, d_2) \leftarrow$ receive
12:     if $from = a_{\mathcal{O}} \wedge phase = \lambda$ then
13:         return ($hk$, "announcement", $d_1$, $d_2$, $content'$, $\{hAnswer_a\}$, $correctness'$, $\{correct_a\}$)
14:     end if
15:     return ($hk$, $phase$, $hContent$, $hCorrectness$, $content'$, $\{hAnswer_a\}$, $correctness'$, $\{correct_a\}$)
16: end procedure
```

Algorithm 2 Announcement phase

```
1: Do nothing.
```

C. Formal Description

The formal description is given in Algorithms 1–6.

D. Validity

The validity of the method is mathematically guaranteed by attributing it to the properties of the cryptographic hash function.

We omit detailed description of validity of the method α due to page limit.

[3]For the simplicity of the mathematical proof, we assume that each $\mathcal{P}_i$ submits their answer only once in the model. In practice, this restriction is unnecessary, and in our proof of concept using Ethereum, which will be described later, an answerer can submit an answer any number of times.

Algorithm 3 Submission start phase

```
1:  procedure $\mathcal{O}(hk, content, correctness)$
2:      send $(a_{\mathcal{C}}, content)$
3:      return $(hk, content, correctness)$
4:  end procedure
5:  procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
6:      $(from, d) \leftarrow$ receive
7:      if $from = a_{\mathcal{O}} \land phase =$ "announcement" $\land \mathrm{Hash}(hk, d) = hContent$ then
8:          return $(hk,$ "submission", $hContent, hCorrectness, d, \{hAnswer_a\}, correctness', \{correct_a\})$
9:      end if
10:     return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
11: end procedure
```

Algorithm 4 Submission phase

```
1:  procedure $\mathcal{P}_i(hk)$
2:      Get $content'$ from the return value of the most recent action of $\mathcal{C}$.
3:      send $(a_{\mathcal{C}}, \mathrm{Hash}(hk, answer_i(content') \| a_{\mathcal{P}_i}))$
4:      return $(hk)$
5:  end procedure
6:  procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
7:      $(from, d) \leftarrow$ receive
8:      if $phase =$ "submission" $\land hAnswer_{from} = \lambda$ then
9:          $\{newHAnswer_a\} \leftarrow \{d \ (a = from),\ hAnswer_a \ (a \neq from)\}$
10:         return $(hk, phase, hContent, hCorrectness, content', \{newHAnswer_a\}, correctness', \{correct_a\})$
11:     end if
12:     return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
13: end procedure
```

Algorithm 5 Judgment start phase

```
1:  procedure $\mathcal{O}(hk, content, correctness)$
2:      send $(a_{\mathcal{C}}, correctness)$
3:      return $(hk, content, correctness)$
4:  end procedure
5:  procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
6:      $(from, d) \leftarrow$ receive
7:      if $from = a_{\mathcal{O}} \land phase =$ "submission" $\land \mathrm{Hash}(hk, d) = hCorrectness$ then
8:          return $(hk,$ "judgement", $hContent, hCorrectness, content', \{hAnswer_a\}, d, \{correct_a\})$
9:      end if
10:     return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
11: end procedure
```

Algorithm 6 Judgment phase

```
1:  procedure $\mathcal{P}_i(hk)$
2:      Get $content'$ from the return value of the most recent action of $\mathcal{C}$.
3:      send $(a_{\mathcal{C}}, answer_i(content'))$
4:      return $(hk)$
5:  end procedure
6:  procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
7:      $(from, d) \leftarrow$ receive
8:      if $phase =$ "judgement" $\land \mathrm{Hash}(hk, d \| from) = hAnswer_{from} \land correct_{from} = \lambda$ then
9:          $\{newCorrect_a\} \leftarrow \{correctness'(d) \ (a = from),\ correct_a \ (a \neq from)\}$
10:         return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{newCorrect_a\})$
11:     end if
12:     return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{correct_a\})$
13: end procedure
```

1) Impossibility of Changing Exam Content: It can be guaranteed that $\mathcal{O}$ cannot change the exam content after the start of the announcement.

2) Impossibility of Changing the Correctness Judgment Programs: It can be guaranteed that $\mathcal{O}$ cannot change the correctness judgment program after the start of the announcement.

3) Impossibility of Changing Answers: It can be guaranteed that $\mathcal{P}_i$ cannot change their answer after submitting it.

4) Validity of the Results of the Correctness Judgment: The above properties provide a corollary that guarantees the validity of the result of the correctness judgments if no sabotage attack is made.

5) Confidentiality of the Exam Content: It can be guaranteed that as long as $\mathcal{O}$ behaves correctly, $\mathcal{P}_i$ cannot know the exam content before the submission start phase.

6) Confidentiality of the Correctness Judgment Programs: It can be guaranteed that as long as $\mathcal{O}$ behaves correctly, $\mathcal{P}_i$ cannot know the correctness judgment program before the judgment start phase.

7) Confidentiality of Answers: It can be guaranteed that a $\mathcal{P}_i$ cannot know the answer of another who behaves correctly before the judgment start phase.

E. Discussion

We discuss the feasibility of applied attacks on the proposed method α and the economic cost when the complexity of the correctness judgment program is large.

1) Replay Attacks by Answerers: It is guaranteed that answerers cannot know the answers of others. However, even if they cannot know the answers of others, they may be able to conduct a replay attack to steal the answers by reproducing the communication of others. Our method is also resistant to replay attacks because answerers do not simply hash their answer but combine it with their address before hashing. This makes it impossible to steal the answers even if an answerer reproduces the communication of others who have different addresses.

2) Length Extension Attacks by Answerers: When answerers know hk and $\mathrm{Hash}(hk, m_1)$, they may be able to compute $\mathrm{Hash}(hk, m_1 \| m_2)$, where m_2 is chosen by them. This attack is called a length extension attack and is known to be effective against MD5 and SHA-1.

In our method, they may be able to steal answers by using a length extension attack. As a countermeasure, we should use SHA-256 or Keccak-256, which are resistant to attacks. In our proof of concept using Ethereum, we use Keccak-256.

3) Sabotage Attacks by Questioners: In our method, $\mathcal{O}$ needs to actively follow the protocol. Therefore, $\mathcal{O}$ can perform a sabotage attack. As a countermeasure, we can make $\mathcal{O}$ pay a deposit in cryptocurrency in advance and confiscate it when $\mathcal{O}$ performs a sabotage attack. In our proof of concept using Ethereum, we implement this feature.

4) When Exam Content or a Correctness Judgment Program is Invalid: A malicious $\mathcal{O}$ may prepare invalid $\mathit{content}$ or $\mathit{correctness}$ and conduct a meaningless exam. The only way to prevent this attack is to perform a prior audit by a trusted authority. However, in our method, $\mathit{content}$ and $\mathit{correctness}$ are disclosed before the answers are disclosed, so it is possible to prevent answers from being given to a malicious $\mathcal{O}$.

5) When the Majority of Block Approvers are Bought Off: Thus far, we have assumed that each transaction is approved after some time. However, it is possible to bribe the majority of block approvers to stop the approval of a transaction.

We believe that this attack is not feasible on major blockchains because the following disadvantages arise when block approvers participate in an attack:

First, if a mining pool participates in an attack, the social credibility of the pool will be lost. In major PoW-based blockchains such as Ethereum, all the major block approvers are mining pools, so this disadvantage cannot be ignored.

Second, if a major block approver participates in an attack, their blockchain loses its public trust. Therefore, the market value of their cryptocurrency will decrease, and block approval rewards will decrease.

Third, the risk of a 51% attack increases because the amount of computational power needed to extend the main chain is reduced during an attempt at this attack; a successful 51% attack results in a loss of public trust in the blockchain.

Some rewards in addition to the above disadvantages must be offered to multiple block approvers. For example, Ethereum requires the acquisition of at least three mining pools [9]. Therefore, we believe that this attack is not feasible.

6) When the Complexity of the Correctness Judgment Program is Large: In the proposed method α, the economic cost of $\mathcal{P}_i$ becomes large when the complexity of $\mathit{correctness}$ is large. This is because the computation of $\mathit{correctness}$ is done on a blockchain. Therefore, we discuss the proposed method β, which incorporates the concept of mutual evaluation.

V. Proposed Method B

The proposed method β reduces the economic cost by mutual evaluation among the answerers instead of a correctness judgment on the blockchain.

A. Intuitive Description

1) (Announcement start phase—Submission phase) These are the same as those of the proposed method α.
2) (Publication start phase) This is the same as the judgment start phase of the proposed method α.
3) (Publication phase) Each $\mathcal{P}_i$ sends $\mathit{answer}_i(\mathit{content}')$ to $\mathcal{C}$ to publish their answer. $\mathcal{C}$ computes the cryptographic hash of the bitstring concatenation of the received data and the source address and verifies that the hash matches the hash recorded in Step 4. If it matches, $\mathcal{C}$ records the source address as $\mathit{participants}_n$ and the received data as $\mathit{answers}'_n$, increments n, and records a uniformly selected value from $\{0, \ldots, n-1\}$ as r.
4) (Peer-review phase) Each $\mathcal{P}_i$ takes j such that $\mathit{participants}_j$ matches their own address and sends $\mathit{correctness}'\left(\mathit{answers}'_{(j+r) \bmod n}\right)$, $\mathit{correctness}'\left(\mathit{answers}'_{(j+r+1) \bmod n}\right)$, and $\mathit{correctness}'\left(\mathit{answers}'_{(j+r+2) \bmod n}\right)$ to C to evaluate the others' answers. $\mathcal{C}$ takes j such that $\mathit{participants}_j$ matches the source address and pushes the received data to $\mathit{correct}_{(j+r) \bmod n}$, $\mathit{correct}_{(j+r+1) \bmod n}$, and $\mathit{correct}_{(j+r+2) \bmod n}$. Finally, whether 2 or more True are pushed to $\mathit{correct}_j$ determines the result of the correctness judgment of the answerer whose address is $\mathit{participants}_j$.
5) (Revision phase) $\mathcal{P}_i$ sends j to $\mathcal{C}$ if the result of the correctness judgment of the answerer whose address is $\mathit{participants}_j$ is wrong. $\mathcal{C}$ revises the result by computing $\mathit{correctness}'(\mathit{answers}_j)$.

B. Formal Description

The formal description is given in Algorithms 7–9. We omit the phases before the publication phase because they are only trivial modifications of the proposed method α.

C. Discussion

We discuss the incentives and the number of people for mutual evaluation. The discussion of the proposed method α in Section IV-E is also applicable to the proposed method β.

1) The Economic Cost: If the mutual evaluation is done properly, the economic cost of $\mathcal{P}_i$ decreases to $O(|\mathit{answer}_i(\mathit{content})|)$ because nothing needs to be done in the revision phase.

2) The Incentives: The discussion thus far does not take into account the incentives for proper mutual evaluation, which may not necessarily reduce the economic cost. As a countermeasure, we can make $\mathcal{P}_i$ pay a deposit in cryptocurrency when they publish their answer and confiscate it when they evaluate others' answers improperly. Furthermore, a $\mathcal{P}_i$ who revises the result of a correctness judgment in the revision

Algorithm 7 Publication phase

```
1: procedure $\mathcal{P}_i(hk)$
2:     Get $content'$ from the return value of the most recent action of $\mathcal{C}$.
3:     send $(a_\mathcal{C}, \mathrm{Hash}(hk, answer_i(content')))$
4:     return $(hk)$
5: end procedure
6: procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{reviewed_j\}, n, r, \{correct_j\})$
7:     $(from, d) \leftarrow$ receive
8:     if $phase =$ “publication” $\wedge \mathrm{Hash}(hk, d \| from) = hAnswer_{from} \wedge from \notin \{participants_j \mid j \in \{0, \ldots, n-1\}\}$ then
9:         $\{newPartcipants_j\} \leftarrow \{participants_j\ (j \in \{0, \ldots, n-1\}),\ from\ (j = n)\}$
10:        $\{newAnswers_j\} \leftarrow \{answer_j\ (j \in \{0, \ldots, n-1\}),\ d\ (j = n)\}$
11:        $\{newReviewed_j\} \leftarrow \{reviewed_j\ (j \in \{0, \ldots, n-1\}),\ \mathrm{False}\ (j = n)\}$
12:        $newN \leftarrow n + 1$
13:        Choose $newR$ uniformly from $\{0, \ldots, n-1\}$.
14:        return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{newParticipants_j\}, \{newAnswers'_j\}, \{newReviewed_j\}, newN, newR, \{correct_j\})$
15:    end if
16:    return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{reviewed_j\}, n, r, \{correct_j\})$
17: end procedure
```

phase can be compensated for the cost of the revision by paying the confiscated cryptocurrency. In our proof of concept using Ethereum, we implement this feature.

3) The Number of People: In the discussion thus far, the number of people for mutual evaluation has been set to 3, but this is an example, and the number is arbitrary in practice.

VI. Proof of Concept

We implemented a decentralized programming contest platform on Ethereum based on the proposed methods α and β. The code is available at https://github.com/azonti/decentcoder. The `main` branch corresponds to the proposed method α, and the `pm2` branch corresponds to the proposed method β.

Below, we propose solutions to the problems that emerged during the implementation process and measure the economic cost of the method using the implementation.

A. How to Distribute the Exam Content

When the proposed method is implemented as is, the exam content is directly recorded on the blockchain. However, the cost of recording data on the Ethereum blockchain is high. Therefore, we implemented the following cost reduc-

Algorithm 8 Peer-review phase

```
1: procedure $\mathcal{P}_i(hk)$
2:     Get $correctness'$, $\{participants_j\}$, $\{answers'_j\}$, $n$, and $r$ from the return value of the most recent action of $\mathcal{C}$.
3:     $\{j'\} \leftarrow \{j \in \{0, \ldots, n-1\} \mid participants_j = a_{\mathcal{P}_i}\}$
4:     send
           $\Big(a_\mathcal{C}, correctness'\big(answers'_{(j'+r) \bmod n}\big), correctness'\big(answers'_{(j'+r+1) \bmod n}\big), correctness'\big(answers'_{(j'+r+2) \bmod n}\big)\Big)$
5:     return $(hk)$
6: end procedure
7: procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{reviewed_j\}, n, r, \{correct_j\})$
8:     $(from, d_1, d_2, d_3) \leftarrow$ receive
9:     if $phase =$ “publication” $\wedge from \in \{participants_j \mid j \in \{0, \ldots, n-1\}\} \wedge from \notin \{reviewed_j \mid j \in \{0, \ldots, n-1\}\}$ then
10:        $\{j'\} \leftarrow \{j \in \{0, \ldots, n-1\} \mid participants_j = from\}$
11:        push $d_1$ to $correct_{(j'+r) \bmod n}$
12:        push $d_2$ to $correct_{(j'+r+1) \bmod n}$
13:        push $d_3$ to $correct_{(j'+r+2) \bmod n}$
14:        $\{newReviewed_j\} \leftarrow \{reviewed_j\ (j \in \{0, \ldots, n-1\} \setminus \{j'\}),\ \mathrm{True}\ (j = j')\}$
15:        return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{newReviewed_j\}, n, r, \{correct_j\})$
16:    end if
17:    return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{reviewed_j\}, n, r, \{correct_j\})$
18: end procedure
```

Algorithm 9 Revision phase

```
1: procedure $\mathcal{P}_i(hk)$
2:     send $(a_\mathcal{C}, j)$
3:     return $(hk)$
4: end procedure
5: procedure $\mathcal{C}(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{reviewed_j\}, n, r, \{correct_j\})$
6:     $(from, d) \leftarrow$ receive
7:     if $phase =$ “publication” $\wedge d \in \{0, \ldots, n-1\}$ then
8:         push $correctness'(answers'_d)$ to $correct_d$ 4 times
9:     end if
10:    return $(hk, phase, hContent, hCorrectness, content', \{hAnswer_a\}, correctness', \{participants_j\}, \{answers'_j\}, \{reviewed_j\}, n, r, \{correct_j\})$
11: end procedure
```

tion method using the InterPlanetary File System (IPFS), a distributed file system proposed by Benet [10]:

1) In the announcement start phase, instead of recording the cryptographic hash of the exam content on the blockchain, we record Advanced Encryption Standard (AES)-encrypted exam content on IPFS and record the IPFS path on the blockchain.
2) In the submission start phase, instead of recording the exam content on the blockchain, the AES passphrase is recorded on the blockchain.

Since the data on IPFS are content-addressed, it is possible to achieve unchangeability similar to that of a cryptographic hash function. However, since anyone who knows an IPFS path can also know the content of the data, we use AES in combination with this method to achieve confidentiality.

B. How to Submit Answers

In the proposed method, when an answerer submits their answer, they submit the cryptographic hash of the bitstring concatenation of their answer and their address. Then, the hash is computed on the blockchain when their answer is published and judged. However, this method requires loading the entire answer into memory, which is costly. Therefore, we implemented the following cost reduction method using the `EXTCODEHASH` instruction proposed and introduced by Johnson and Bylica [11]:

1) In the submission phase, the answerers submit the hash of the bitstring concatenation of their answer's hash and their address. In other words, their answer is hashed twice.
2) In the publication and judgment phase, their answer is deployed on the blockchain as a contract. In the blockchain, the `EXTCODEHASH` instruction is issued to compute the hash of the answer, which is then combined with the source address and hashed again.

C. How to Check the Purity of Programs

The proposed method requires that correctness judgment programs be pure; i.e., programs must not change their behavior according to the environment. Otherwise, some judgment results will not be uniquely determined.

However, the Ethereum virtual machine has instructions to read values from the environment[4], which must be prohibited.

It may seem sufficient to scan the instructions in a program one by one to make sure that there are no instructions that should be prohibited. However, since there are also data areas in a program, interpreting all data areas as instructions will result in false positives with nonnegligible probability. We devised the following method to reduce the probability of false positives to a practical level:

1) Scan the instructions in a program one by one from the beginning. If an instruction that should be prohibited is encountered, go to step 2. If an invalid opcode, `STOP`, `JUMP`, `RETURN`, `REVERT`, or `SELFDESTRUCT` is encountered, go to Step 3.
2) If `JUMP`, `JUMPI`, `RETURN`, or `REVERT` is encountered, determine that the program is nonpure. If an invalid opcode, `STOP`, or `SELFDESTRUCT` is encountered, go to step 3.
3) If `JUMPDEST` is encountered, go to Step 1.

The validity of this method is explained as follows: Only areas from the beginning or `JUMPDEST` to invalid opcodes, `STOP`, `JUMP`, `RETURN`, `REVERT`, or `SELFDESTRUCT` will be executed. Therefore, only such areas need to be monitored. Additionally, even if a prohibited instruction is executed, it will be harmless because environment-dependent values will be discarded unless `JUMP`, `JUMPI`, `RETURN`, or `REVERT` are executed afterwards. Hence, we only need to detect such situations.

[4]`ADDRESS`, `BALANCE`, `ORIGIN`, `CALLER`, `GASPRICE`, `EXTCODESIZE`, `EXTCODECOPY`, `EXTCODEHASH`, `BLOCKHASH`, `COINBASE`, `TIMESTAMP`, `NUMBER`, `DIFFICULTY`, `GASLIMIT`, `SLOAD`, `GAS`, `CALL`, `CALLCODE`, `DELEGATECALL`, and `STATICCALL`

Table I
ECONOMIC COST

Method	For a Questioner (USD)	For an Answerer (USD)
α	38.66	48.36
β	49.99	33.56

D. Economic Cost Evaluation

We evaluated the economic cost of this implementation. The data used to organize the pseudocontest for the evaluation are available at https://github.com/azonti/dbc001. Table I shows the evaluation results. The gas price, which indicates the amount of cryptocurrency per instruction, was set to 25×10^{-9} ETH, and 1 ETH was set to 601.4 USD. The former is the median gas price of newly approved transactions on December 6, 2020 [12], and the latter is the closing price of Bibox, a cryptocurrency exchange, on December 6, 2020 [13].

E. Future Instruction Implementations Will Lower the Economic Cost

As mentioned earlier, this implementation guarantees the purity of correctness judgment programs by scanning the instructions. However, when the instructions below are implemented in the future, there will be no need to scan programs, and the economic cost will decrease.

First, `PURE_CALL` is needed. Programs invoked through this instruction cannot execute instructions that read values from the environment. This instruction was proposed by Buterin [14] in 2017 but has not yet been implemented.

Second, `SANDBOXED_CALL` is required. Programs called through this instruction cannot call any other program. This instruction was proposed by Zoltu [15] in 2016 but has not yet been implemented.

Once the above instructions are implemented, we can prohibit the virtual machine from reading values from the environment. Thus, the need to scan programs is eliminated, and the economic cost is reduced.

Table II
ECONOMIC COST
WHEN `PURE_CALL` AND `SANDBOXED_CALL` ARE IMPLEMENTED

Method	For a Questioner (USD)	For an Answerer (USD)
α	37.83	25.14
β	49.16	10.59

We estimated the economic cost of implementing these instructions. The data used for the estimation are the same as in Section VI-D. Table II shows the estimation results.

VII. CONCLUSION

We proposed a decentralized examination protocol that can handle a set of correct answers defined by a program. The validity of the protocol, which consists of unchangeabilities and confidentialities, is mathematically guaranteed by attributing it to the properties of cryptographic hash functions. We also studied the feasibility of applied attacks such as replay attacks and length extension attacks and proposed countermeasures against such attacks.

We also implemented a decentralized programming contest platform based on the proposed method on Ethereum and demonstrated the feasibility of the method. In addition, we proposed solutions to the problems that emerged during the implementation process and measured the economic cost of the method using the implementation. As a result, we confirmed that the method is feasible for approximately 30–40 USD.

REFERENCES

[1] J. Williams, "Who is Rick Singer? The key CEO helped Lori Loughlin, Felicity Huffman daughters and plenty more wealthy scam their way into college," https://www.newsweek.com/rick-singer-college-admissions-scam-1384647, Apr. 2019, Accessed: Aug. 1, 2021.

[2] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, Oct. 2008.

[3] "Namecoin," https://www.namecoin.org/.

[4] Y. Yoshimura, "New capture the flag (CTF) using Bitcoin," https://qiita.com/yyu/items/b6f367eb876dd28e759a, Jan. 2018, Accessed: Dec. 10, 2020 (in Japanese).

[5] K. Sawada, "CTF with prize using dapps," https://narusejun.com/archives/26/, Dec. 2018, Accessed: Dec. 10, 2020 (in Japanese).

[6] Y. Kaneko, S. Tanaka, T. Kimura, J. Okumura, S. Osada, S. Azuchi, "A decentralized examination system using public blockchain technologies," IEICE Tech. Rep., 118(480), pp.233–238, Mar. 2019 (in Japanese).

[7] "Ethereum whitepaper," https://ethereum.org/en/whitepaper/, Oct. 2020, Accessed: Dec. 10, 2020.

[8] V. Buterin, "On public and private blockchains," https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/, Aug. 2015, Accessed: Dec. 10, 2020.

[9] "Ethereum hash distribution," https://blockchair.com/ethereum/charts/hashrate-distribution.

[10] J. Benet, "IPFS — Content addressed, versioned, P2P file system," arXiv:1407.3561v1 [cs.NI], Jul. 2014.

[11] N. Johnson, P. Bylica, "EIP-1052: EXTCODEHASH opcode," Ethereum Improvement Proposals, no.1052, May. 2018.

[12] "Ethereum median gas price chart," https://blockchair.com/ethereum/charts/median-gas-price.

[13] "ETH USD Bibox historical data," https://www.investing.com/crypto/ethereum/eth-usd-historical-data.

[14] V. Buterin, "Create pure_call.md," https://github.com/ethereum/EIPs/pull/195, Jan. 2017.

[15] M. Zoltu, "New opcodes: SANDBOXED_CALL," https://github.com/ethereum/EIPs/issues/117, Jun. 2016.