Incentive Distribution Scheme for Digital Publishers Based on Smart Contracts

Yuhuan Hu[1], Kai Zhao[1], and Linlin Zhang[2]

107552103713@stu.xju.edu.cn, zhawkk@xju.edu.cn, zllnadasha@xju.edu.cn

[1] College of Computer Science and Technology, Xinjiang University, Urumqi, China

[2] College of Software, Xinjiang University, Urumqi, China

Abstract. Smart contracts, as a type of contract technology on the blockchain, can effectively address many shortcomings of traditional contract systems in digital environments, including opaque contract execution, regulatory difficulties, and low dispute resolution efficiency, provide publishers with a more efficient, transparent, and trustworthy contract solution. This paper takes academic journal publishers as an example and proposes an efficient incentive allocation smart contract. This paper firstly outlines the issues present in the academic journal submission process, then provides a detailed description and introduction of the proposed submission process based on smart contracts. It explains how it effectively alleviates the shortcomings of traditional contract systems and protects identity privacy.Finally, experiments are conducted on the gas cost and latency of the smart contract proposal. The experiments indicate that the proposed solution has certain economic and efficient characteristics.

Keywords: Smart Contracts, Incentive Distribution, Blockchain, Privacy Protection

1 Introduction

Digital publishing is the process of creating, disseminating, and distributing publications such as books, magazines, newspapers, etc., utilizing digital technology and internet platforms. With the advancement of digitization, digital publications like e-books [1], online journals [2], digital news [3], etc., are gradually becoming the mainstream forms of publishing. Digitization has not only altered the production, dissemination, and consumption patterns of traditional publications but has also profoundly impacted various aspects of the publishing industry such as industrial structure, business models, market competition, copyright protection, among others [4].

However, despite the numerous conveniences and opportunities brought forth by digital publishing, it also faces a series of contractual disputes. With the continuous progress and application of digital technology, the terms and agreements of publishing contracts need to adapt to ensure effective protection of the rights and interests of all parties involved. First of all, digital publishing involves collaboration among multiple parties such as authors, publishers, digital platforms, etc., thus the allocation of rights, responsibilities, and distribution of interests within contracts often becomes contentious. For instance, copyright issues in digital publishing, including ownership, transfer, and scope of digital rights, frequently emerge

ICBBEM 2024, March 29-31, Wuhan, People's Republic of China
Copyright © 2024 EAI
DOI 10.4108/eai.29-3-2024.2347447

as core aspects of contractual disputes. Secondly, digital publishing involves cross-platform and cross-border distribution and sales, necessitating clear provisions within contracts regarding content licensing, distribution channels, sales models, etc., to avoid disputes arising from geographical differences or platform characteristics. Additionally, with the ease of replication, dissemination, and modification of digital content, digital publishing contracts also require stricter protective measures to prevent unauthorized content usage, infringement, etc.

Blockchain technology presents novel prospects for the digital publishing sector, enabling transparency, security, and efficiency in contractual processes [5]. Through features such as transparent record-keeping, smart contracts, and asset tokenization, blockchain revolutionizes traditional publishing practices, fostering innovation and collaboration among stakeholders while safeguarding rights and interests in a decentralized ecosystem.Smart contracts, as an application of blockchain technology, further strengthen the efficiency and reliability of contract management in the digital publishing sector [6]. Smart contracts are self-executing contracts coded in a programming language, capable of automatically executing their terms when predefined conditions are met, and are immutable. In the realm of digital publishing, smart contracts can be employed to automatically execute contract terms related to digital copyright authorization, distribution, and sales, ensuring the timely and accurate execution of rights for all parties involved, thereby reducing disputes arising from contract execution issues. Moreover, smart contracts can enhance the transparency and fairness of multi-party contract management, thereby bolstering the credibility of contract execution and reducing the probability of contract disputes [7].

This paper proposes the design and implementation of a smart contract system using the peer review process of academic journals as a case study to address copyright protection and contract execution issues within digital publishing houses. The aim is to enhance the transparency, efficiency, and fairness of the peer review process, thereby promoting the digital publishing industry towards a more efficient, transparent, and secure digital transformation.

2 Risks in the Academic Journal Submission Process

The submission process for academic journals typically follows a structured series of steps designed to ensure the quality and integrity of published research [8], as show in **Fig.1**. Authors prepare their research manuscripts according to the submission guidelines provided by the journal. This includes formatting the manuscript according to the journal's style, adhering to word limits, and providing necessary documentation such as figures, tables, and references.

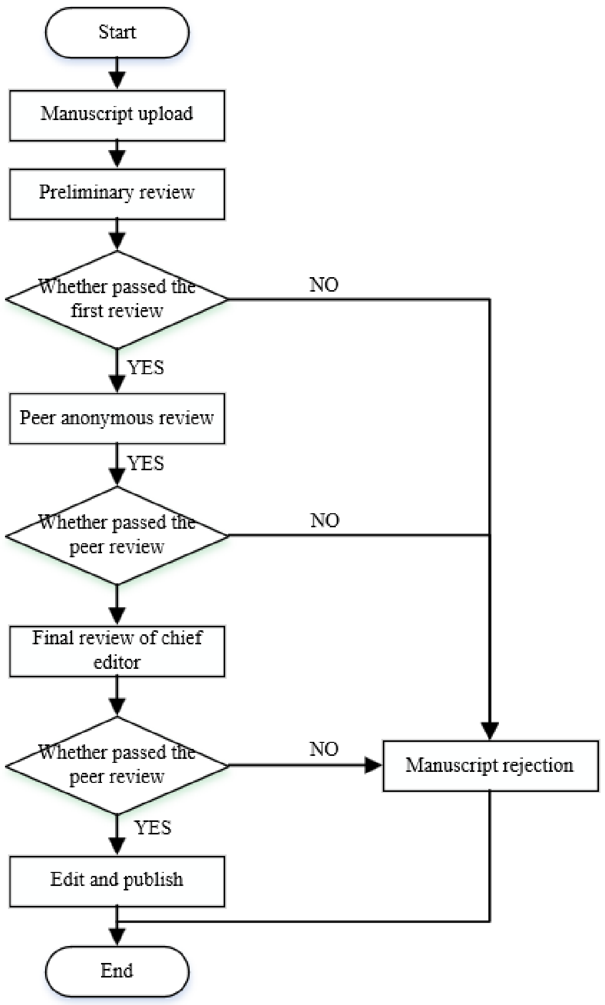
Start
Manuscript upload
Preliminary review
Whether passed the first review
NO
YES
Peer anonymous review
YES
Whether passed the peer review
NO
Final review of chief editor
Whether passed the peer review
NO
Manuscript rejection
YES
Edit and publish
End

Fig. 1. Academic journal submission process.

Authors submit their manuscripts electronically through the journal's online submission system. This usually involves creating an account on the journal's submission platform, inputting metadata about the manuscript, such as title, abstract, keywords, and uploading the manuscript file. Upon submission, the manuscript undergoes an initial screening by the journal's editorial team. This screening may check for adherence to submission guidelines, appropriateness of the topic for the journal's scope, and overall quality of the manuscript. Manuscripts that do not meet the journal's criteria may be rejected at this stage without peer review. In the submission process, there may be risks of data leakage or data tampering.

Manuscripts that pass the initial screening are typically sent out for peer review. Peer review involves sending the manuscript to external experts (peer reviewers) in the field who evaluate the quality, validity, and significance of the research. Reviewers provide feedback on the manuscript's strengths, weaknesses, and suitability for publication. Based on the reviewers' comments, the editor makes a decision on whether to accept, reject, or request revisions to the manuscript. However, if the journal fails to effectively address intellectual property issues between authors and reviewers, it may lead to intellectual property disputes, posing unnecessary legal risks and costs for both the journal and the involved researchers.

If revisions are requested, authors revise their manuscripts according to the reviewers' comments and submit the revised version along with a response letter detailing the changes made in response to the reviewers' feedback.

The editor evaluates the revised manuscript and decides whether it is suitable for publication based on the reviewers' comments, the authors' revisions, and the journal's editorial criteria. The editorial decision may include acceptance, rejection, or further revisions. Accepted manuscripts undergo copyediting, typesetting, and proofreading to ensure consistency, clarity, and adherence to journal style. Once finalized, the manuscript is published either online, in print, or both, depending on the journal's publication model. Subsequently, traditional publishers often require a considerable amount of time to process payments, which may lead to errors or delays.

3 Incentive Distribution Approach Based on Smart Contracts

3.1 Legal Framework for Smart Contracts

The primary task in establishing a legal regulatory framework for smart contracts is to clarify their legal status within the legal system. Smart contracts can largely be equated with traditional contracts, clearly defining the parties involved, specifying rights and obligations, and automatically executing contract terms under specific conditions. For instance, smart contracts can be written using Ethereum smart contract languages, explicitly stating the identities of the parties, their rights and obligations, as well as the conditions and rules for contract execution. Smart contracts can be deployed on blockchain networks to ensure their immutability and enforceability. Therefore, smart contracts can, to a certain extent, be treated equivalently to traditional contracts and be explicitly regulated within a legal framework.

Regulatory bodies play a crucial role in the design and execution process of smart contracts. These bodies are responsible for reviewing the design and execution of smart contracts to ensure their legality and compliance. Establishing specialized agencies or committees can facilitate the review and supervision of the design and execution process of smart contracts. These bodies can provide legal consultation and guidance services and impose penalties and sanctions on illegal activities if necessary to ensure the safety and reliability of smart contract transactions.

To address disputes that may arise during the execution of smart contracts, an effective dispute resolution mechanism should be established. Initially, parties can attempt to resolve disputes through negotiation. If consensus cannot be reached, the contract may include arbitration clauses, allowing parties to choose an independent arbitration institution for resolution. Additionally, if the contract does not include arbitration clauses, parties can choose to submit disputes to the court for resolution. Throughout the dispute resolution process, it is essential to ensure fairness, transparency, and respect for the provisions agreed upon in the contract to uphold the effective execution of the contract.

3.2 Smart Contract Design

The incentive allocation smart contract is used to manage the journal submission process, including expert review, final acceptance or rejection, and related fee payments. This smart contract not only simplifies the journal submission process but also provides participants with increased security and transparency. By managing manuscript and reviewer information through structs, data integrity and confidentiality are ensured. Automatic handling of review results and layout fee payments not only saves time but also reduces the risk of human error. Additionally, through the mapping functionality of the smart contract, each step can be tracked and recorded, thereby enhancing the transparency and traceability of the process.

The contract includes a series of functions and events for executing and recording important operations and state changes in the contract, enabling external observers to monitor and respond to events, as show in **Table 1**. The constructor initializes the contract, the *addReviewer* function allows the journal institution to add reviewers, the *completeReview* function enables reviewers to complete the review of manuscripts and submit review results, and the *finalizeManuscript* function allows the journal institution to end manuscript review and handle manuscripts based on review results. The *ReviewCompleted* event is triggered when a reviewer completes the review of a manuscript, recording the reviewer's address, the timestamp of completing the review, and the review result, allowing the journal institution to understand the reviewer's review result. The *ManuscriptRes* event is triggered when the journal makes final decisions on digital manuscripts, recording the author's account address and the timestamp of acceptance or rejection of the manuscript. The *ReviewerAdded* event is triggered when the journal adds a new reviewer, recording the account address of the newly added reviewer and the corresponding manuscript address. The *FeesPaid* event is triggered when the author pays fees, used to record the payment status of fees.

Table 1. Contract functions and events.

Type	Name	Function
funcation	constructor	Initializes the contract deployer, base fee, and expert review fee.
	addReviewer	Adds a reviewer, which can only be called by the journal organization.
	completeReview	Allows a reviewer to complete the review process and set the review result.
	finalizeManuscript	Allows the journal organization to finalize the manuscript review based on the review results.
	payAllFees	Allows the author to pay all fees, including base fee and expert review fees, and distribute payments to reviewers.
event	ReviewCompleted	Triggered when a reviewer completes the review, emitting the reviewer's address, timestamp, and review result.
	ManuscriptRes	Triggered when a manuscript is finalized, emitting the author's address, timestamp, and final review result.
	ReviewerAdded	Triggered when a reviewer is added, emitting the reviewer's address and manuscript address.
	FeesPaid	Triggered when all fees are paid, emitting the payer's address and the amount paid.

3.3 Smart Contract Solutions for Traditional Contract Limitations

Before implementing the incentive allocation contract, deployment and initialization of the contract are necessary. The constructor takes three parameters: the author's address, the basic layout fee, and the expert review fee. Within the constructor, *author* is assigned to the contract's author variable, enabling subsequent operations of the contract to identify and record the author's account address, facilitating account transactions. *baseFee* represents the layout fee set by the journal institution. Once the review process is completed, subsequent payment functions will automatically execute, deducting the corresponding layout fee from the author's account address. Similarly, *expertReviewFee* denotes the fee that reviewers will receive for reviewing manuscripts.

The journal institution utilizes funcation *addReviewer* to add reviewers and assign submitted manuscripts to them. This algorithm can only be invoked by the journal institution and is used for adding new reviewers. Initially, it checks if the reviewer already exists. If not, the reviewer is added to the reviewer mapping, recording relevant information including manuscript address, review status, and submission time. Subsequently, the status of the manuscript is adjusted to under review. Upon completion of adding reviewers by the journal institution, a *ReviewerAdded* event is sent, notifying external observers of the corresponding reviewer and the manuscript assigned to them for review. This function ensures the privacy protection of reviewers.

```
Function addReviewer (address _reviewer, address _manuscript) external onlyPublisher {
        require(!reviewers[_reviewer].exists, "exists");
        reviewers[_reviewer].exists = true;
        reviewersAddresses.push(_reviewer);
        manuscripts[_manuscript].manuscriptAddress=_manuscript;
        manuscripts[_manuscript].result= ReviewResult.Pending;
        manuscripts[_manuscript].submittedAt = now;
        emit ReviewerAdded(_reviewer, _manuscript);
}
```

After the assigned expert completes the evaluation of the manuscript, funcation *completeReview* is invoked to establish the review outcome and document it. Initially, the algorithm verifies whether the caller qualifies as an eligible reviewer and has not concluded the review process. Subsequently, it determines the review result and flags the reviewer's completion of the evaluation. Ultimately, it initiates the *ReviewCompleted* event to log the reviewer's actions and the outcome of the review. This function can only be invoked by designated peer reviewers.

```
function completeReview(ReviewResult _result) external {
        require(reviewers[msg.sender].exists, "Not assigned");
        require(reviewers[msg.sender].reviewed==false,"Finshed");
        require(_result!= ReviewResult.Pending, "Invalid result");
        reviewers[msg.sender].result = _result;
        reviewers[msg.sender].reviewed = true;
        emit ReviewCompleted(msg.sender, block.timestamp,_result);
}
```

After all reviewers have completed the review process, the journal institution invokes function *finalizeManuscript* to determine whether to accept the manuscript based on the opinions of the invited reviewers. This function can only be called by the journal institution. Initially, it verifies if the final review outcome is valid, ensuring that the final outcome can only be acceptance or rejection. Then, it proceeds based on the different outcomes. If the manuscript is accepted, the author is required to pay the publication fee set by the journal institution. If the manuscript is rejected, only 10% of the publication fee is calculated. Upon completion of the calculation, a *ManuscriptRes* event is triggered to record the paid.

```
  function finalizeManuscript(address _manuscript, ReviewResult _finalResult) external
  onlyPublisher {
        require(_finalResult==ReviewResult.Accepted||_finalResult
        ==ReviewResult.Rejected, "Invalid final result");
        Manuscript storage manuscript = manuscripts[_manuscript];
       if (manuscript.submittedAt != 0) {
         if (_finalResult == ReviewResult.Accepted) {
           totalFees += baseFee;
             } else {
           uint rejectionFee = baseFee * 10 / 100;
           totalFees += rejectionFee;
      }
        emit ManuscriptRes(author, block.timestamp,_finalResult);
        manuscript.result = _finalResult;
        } else {
       revert("Manuscript not found");
        }
}
```

Once a manuscript has been accepted or rejected by the journal institution, the author is required to pay publication fees as well as the fees for expert reviews through function *payAllFees*. This function can only be invoked by the author. To begin wirh, it checks whether the caller has submitted a manuscript. Then, it calculates the total fees, including publication fees and all review fees. Subsequently, it ensures that the payment amount equals the total fees and proceeds to pay the review fees to each reviewer. Finally, it triggers a *FeesPaid* event to record the payment status.

```
  function payAllFees() external payable {
        require(manuscripts[msg.sender].submittedAt != 0, "No manuscript submitted by the
        caller");
        uint totalPayment = baseFee + totalReviewFees * reviewersAddresses.length;
        require(msg.value==totalPayment, "Incorrect fee amount");
        for (uint i = 0; i < reviewersAddresses.length; i++) {
       address reviewerAddress = reviewersAddresses[i];
       reviewerAddress.transfer(expertReviewFee);
        }
        totalFees += totalPayment;
        emit FeesPaid(msg.sender, msg.value);
}
```

3.4 Privacy Protection Methods in The Review Process

To achieve secure storage and transmission of digital manuscripts, this paper combines decentralized storage and encryption technologies, and records corresponding information and access permissions on the blockchain.

ECC (Elliptic Curve Cryptography) is a highly secure and efficient encryption algorithm that can provide security comparable to RSA but with shorter key lengths. Journal institutions generate ECC key pairs during system initialization, including a private key securely stored by the journal institution and a corresponding public key distributed to users for encryption operations. Upon registration, users are provided with an ECC public key generated by the journal institution.

During the transmission of digital manuscripts, the manuscripts are first uploaded to IPFS. IPFS is a decentralized file storage system that ensures file availability and persistence by distributing files across multiple nodes in the network. This effectively disperses data storage, reduces the risk of single-point failures, and increases data reliability. After uploading to IPFS, a unique IPFS address is generated for each file, enabling the location of the corresponding file within the IPFS network. This address serves as the identifier for the file, facilitating recording and referencing on the blockchain.

Subsequently, IPFS (InterPlanetary File System) is encrypted using ECC public keys to ensure the security and privacy of files during transmission. By encrypting files with the recipient's public key, only users holding the corresponding private key can decrypt the files, thereby protecting the content from unauthorized access.

Similar to the manuscript circulation process, expert reviews and recommendations for digital manuscripts are also uploaded to IPFS and encrypted using ECC public keys before being uploaded to the blockchain, ensuring the security and integrity of expert reviews and recommendations for digital manuscripts.

On the blockchain, user transactions are typically conducted using addresses rather than directly using user identity information. Each user has addresses generated by key pairs, which are unrelated to user identity information. When users conduct transactions, they sign transactions with their private keys and broadcast them to the blockchain network. Other users can verify the validity of transactions using public keys, but do not know the actual identities behind these addresses. This approach protects user identity privacy.

4 Evaluation and Results

4.1 Smart Contract Gas Consumption

In this paper, the consumption assessment of smart contract gas is realized through Ethereum test platform Remix. Remix is an Ethereum test platform used for conducting assessments of smart contract gas consumption. The gas cost result is shown in **Table 2**. Gas is the unit used in Ethereum to measure the computational effort required to execute a single or a group of actions. Gwei, on the other hand, is a denomination of Ethereum, defined as 1 gwei = 0.000000001 ETH, with one Ether comprising one billion gwei. According to CoinMarketCap's data on April 24, 2023, 1 Ether is approximately equal to 1850 US dollars.

Table 2. Smart contract gas consumption.

Function call	gwei consumption	ETH consumption	USD
Contract deploy	1102005	0.001102005	2.04
Reviewer addition	115898	0.000115898	0.21
Review completion	8426	0.000008426	0.02
Result confirmation	18169	0.000018169	0.03
Fee payment	20306	0.000020306	0.04

In the incentive allocation contract, the gas cost for contract deployment is the highest at 1102005 gwei, approximately $2.04 USD. The gas costs for reviewer addition, review completion, result confirmation, and fee payment are relatively lower, at 115898 gwei, 8426 gwei, 18169 gwei, and 20306 gwei, respectively. Generally, the reviewer addition and confirmation occur fewer than 10 times, and result confirmation and fee payment are executed only once. The maximum expenditure of this contract does not exceed $5 USD, rendering the overall cost of the manuscript workflow reasonable and acceptable. In conclusion, the gas consumption of the proposed smart contract design is rational and meets the practical application requirements.

4.2 Smart Contract Latency Test

Latency refers to the duration between the initiation of a contractual operation or transaction and its finalization and execution on the blockchain. Typically measured in seconds, this metric encapsulates the time required for various processes inherent to blockchain operations, encompassing transaction dissemination, block formation, consensus mechanism validation, and the execution of contract code.

For this experiment, the infrastructure comprises an 11th Gen Intel Core i5-1135G7 processor, boasting a 2.40 GHz base frequency, featuring 4 cores and 8 logical processors, alongside 16.0 GB of physical memory. The experimental setup entails the deployment of a blockchain prototype system, leveraging Fabric 1.4.4, operating within an Ubuntu 18.04 (64-bit) virtual machine configuration with an 8-core processor and 16 GiB of memory. The evaluation of smart contracts is facilitated through Caliper, a comprehensive toolkit and open standard tailored for tracking and assessing blockchain performance. Caliper streamlines the process by providing a standardized methodology for quantifying and contrasting performance metrics across diverse blockchain platforms, thereby empowering developers and enterprises to gauge the efficiency and scalability of their chosen blockchain solutions.

The experiment adopts a deployment model of single machine with multiple nodes to conduct latency testing on smart contracts using Caliper, while varying the number of transactions sent per second. The latency of the incentive allocation contract under different TPS (Transactions Per Second) is illustrated as Fig. 2.

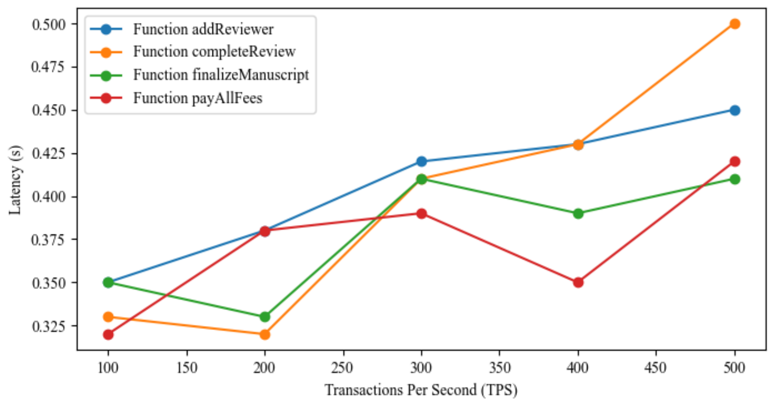

Fig. 2. The latency of smart contracts under different TPS.

Based on the provided experimental data analysis, the functions of the smart contract exhibit relatively stable performance with minor variations in execution time across different throughput levels. The functions for adding reviewers, completing reviews, confirming results, and paying fees show no significant trends in their execution times, indicating the contract can maintain a stable performance level as throughput increases. Therefore, the smart contract demonstrates good performance in terms of efficiency.

4.3 Smart Contract Access Control

To test the access control of smart contracts, this study utilized the jvassdk provided by the Fabric blockchain platform to design three types of identities within the platform: journal institution, digital manuscript author, and peer reviewer. Subsequently, based on the different identities, various smart contract functions were invoked to obtain experimental results, as shown in **Table 3**.

Table 3. Smart contract function access condition.

Function call	Journal institution	Digital manuscript author	Peer reviewer
Contract deploy	YES	NO	NO
Reviewer addition	YES	NO	NO
Review completion	NO	YES	NO
Result confirmation	YES	NO	NO
Fee payment	NO	NO	YES

The journal institution is allowed to deploy contracts, add peer reviewers, and confirm results, but cannot complete reviews. Digital manuscript authors are permitted to complete reviews but cannot deploy contracts, add peer reviewers, or confirm results. Peer reviewers can add other reviewers and complete reviews but cannot deploy contracts, confirm results, or make payments. This access control mechanism helps ensure the security and compliance of transactions and operations conducted on the blockchain platform.

4.4 Digital Manuscript Upload Download Latency Test

Considering that the IPFS network is utilized for storing and retrieving a vast amount of data, where data upload and download requests arrive at network nodes in the form of transactions at varying TPS, different TPS values represent the network's ability and speed in processing requests. Thus, this study selected a 10MB file size and conducted tests on the time overhead of IPFS upload and download under different throughput scenarios. The test results are illustrated in **Fig.3**.

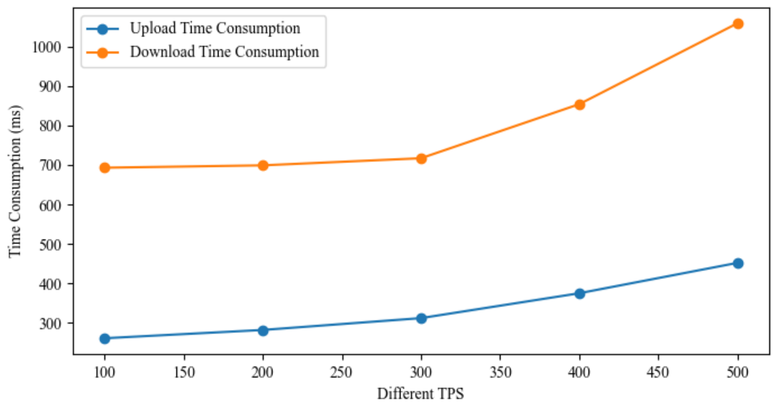

Fig. 3. The latency of IPFS upload and download under different TPS.

As the TPS increase, the upload and download times of the files exhibit a relatively stable trend overall. Although there are slight fluctuations, it can be observed that, in general, with the increase in TPS, the performance of both upload and download remains relatively stable, without significant fluctuations or abrupt changes.

5 Conclusions

The digital publishing industry, driven by technological advancements, has transitioned towards digital formats, necessitating robust contractual frameworks to address emerging challenges.The proposed smart contract, designed with the peer review process of academic journals as a case study, addresses key issues within digital publishing, including copyright safeguard, uer privacy protection, and contract execution. Through transparent and automated execution of contract terms, the system aims to foster transparency, efficiency, and fairness in the digital publishing ecosystem. In conclusion, the integration of blockchain technology, particularly smart contracts, offers promising solutions to address contractual challenges in the digital publishing industry. By leveraging decentralized and transparent mechanisms, smart contracts enhance trust, efficiency, and security, paving the way for a more seamless and equitable digital publishing ecosystem.

Acknowledgments. This study is funded by National Natural Science Foundation project (No. 62366052). This study is one of the phased achievements of the fund.

References

[1] Steiner A. The global book: micropublishing, conglomerate production, and digital market structures[J]. Publishing Research Quarterly, 2018, 34: 118-132.
[2] Hsu D H, Hsu P H, Zhao Q. Rich on paper? Chinese firms' academic publications, patents, and market value[J]. Research Policy, 2021, 50(9): 104319.
[3] Perreault G P, Ferrucci P. What is digital journalism? Defining the practice and role of the digital journalist[J]. Digital Journalism, 2020, 8(10): 1298-1316.
[4] Hokkanen H, Walker C, Donnelly A. Business Model Opportunities in Brick and Mortar Retailing Through Digitalization[J]. Journal of Business Models, 2020, 8(3): 33-61.
[5] Xu M, Chen X, Kou G. A systematic review of blockchain[J]. Financial Innovation, 2019, 5(1): 1-14.
[6] Zou W, Lo D, Kochhar P S, et al. Smart contract development: Challenges and opportunities[J]. IEEE transactions on software engineering, 2019, 47(10): 2084-2106.
[7] Green S, Sanitt A. The Contents of Commercial Contracts: Smart Contracts[J]. The Contents of Commercial Contracts: Terms Affecting Freedoms, Forthcoming, 2019.
[8] Kichloo A, Albosta M, Koul H, et al. Current challenges for researchers during the process of submission and publication[J]. Postgraduate medical journal, 2022, 98(1160): 405-407.