Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3274454, 9 pages
https://doi.org/10.1155/2022/3274454

Research Article

Form Specification of Smart Contract for Intellectual Property Transaction Based on Blockchain

Zhihao Duan, Wenlong Feng, Wang Zhong, Mengxing Huang, and Siling Feng

School of Information and Communication Engineering, Hainan University, Haikou 570228, China

Correspondence should be addressed to Wenlong Feng; fwlfwl@163.com

Received 12 May 2022; Accepted 2 August 2022; Published 21 August 2022

Academic Editor: Jun Cai

In view of the current chaotic structure of smart contracts and no unified definition of smart contracts, the implementation of smart contracts for the same business in the same field is quite different due to different development institutions and operating platforms, resulting in a low level of smart contract sharing and high development costs, hindering the development of smart contracts. Combined with the intellectual property transaction scenario, this paper proposes a blockchain-based intellectual property transaction smart contract form specification, designs the overall structure and process specification of the smart contract in the intellectual property transaction process, formulates a standard for the standardization of smart contract writing in intellectual property transaction scenarios, and solves the current chaotic structure of intellectual property transaction smart contracts, which is conducive to the collaborative development of scholars in various fields.

1. Introduction

Since the introduction of Bitcoin in 2008, the concept of blockchain has rapidly gained popularity. One of the prominent uses is to enable smart contracts [1]. Smart contracts were first proposed by Szabo [2] in 1995 and have a history of more than 20 years. When they were proposed, smart contracts were defined as "a smart contract is a set of commitments defined in digital form, including agreements on which contract participants can execute these commitments," which was originally designed to act as a common agent for both parties as a computer transaction term for executing contracts without a third party and efficient and secure performance of the contract.

Existing smart contract languages, such as Solidity [3] and Hawk [4], are usually specified based on the IT industry, and it is difficult for non-IT disciplines to understand smart contracts. At present, the implementation methods of smart contracts are quite different. The vague definition of smart contracts and different implementation methods have caused the public to have difficulty in understanding smart contracts, thus hindering the development of smart contracts.

At present, many scholars and enterprises in the society are actively involved in smart contract research, such as Google, Microsoft, Facebook, and Amazon [5], and other internationally renowned companies have begun to accelerate the research on blockchain technology and launch solutions and applications. The global blockchain industry has ushered in explosive development and has developed a series of applications in financial services, supply chain management, smart manufacturing, social welfare, and healthcare [6].

The literature [7] pointed out that the current smart contract formalization technologies mainly include theorem proving, symbolic execution, and model checking. These technologies are mainly used to verify the functional correctness of smart contracts. The literature [8] proposed a specification language for smart contracts, SPESC, which defines the specification of smart contracts for collaborative design, which can use a natural language-like syntax to clearly define the obligations and rights of all parties and the transaction rules of cryptocurrencies. The literature [9] provides a framework for automatically generating smart contracts that enable seamless translation of constraint-encoded knowledge representations as requirements of the blockchain. Domain-specific knowledge is encoded using ontology and

semantic rules, and then the structure of the abstract syntax tree is used to incorporate the required rules. The literature [10] describes a fully functional framework and system for specifying and executing smart contracts and includes a browser-based rule smart editor, a compiler that supports a wide variety of rule syntax, and code generation that maps to a RETE-based rule engine. The literature [11] proposed a new language design TA-SPESC for asset-driven specific smart contracts to effectively support asset transactions. And a series of generation rules are proposed to convert TA-SPESC contracts into executable contract procedures. The literature [12] analyzed and compared the existing smart contracts, gave a formal definition of contract-oriented smart contracts, and proposed a general smart contract implementation method independent of the blockchain platform. The literature [13] proposed a logic-based smart contract. The logic statement can be understood as the execution specification of the smart contract, thereby reducing the risk of smart contract errors. The logic-based method can reduce the execution cost and efficiency of smart contracts. The literature [14] proposed a formal method of model checking to verify the security of interactive smart contracts developed and controlled for different entities. Use the NuSMV Model Checker and Behavior Interaction Prioritization tool to model the behavior of smart contracts and their interactions to verify that smart contracts meet the requirements of system functionality. The literature [15] proposed a new design scheme of blockchain copyright management system based on digital watermark, which uses digital watermark and perceptual hash function to generate hash value to ensure the legitimate rights and interests of authors. The literature [16] designed a smart contract model for technology service transaction based on consortium chain according to the technology service transaction scenario, to solve the problems of high trust cost and low default cost in traditional technology service transaction.

To sum up, the current research on smart contracts mainly focuses on security issues and the formalization of smart contracts for code security. The research on the formal specification of smart contracts for intellectual property transactions is rarely involved. Therefore, this paper discusses the existing problems of traditional intellectual property transactions, formulates a set of smart contract formal specifications for intellectual property transactions, and uses a unified smart contract formal model to describe the intellectual property transaction process. It can enable the public to better understand the relevant definitions of smart contracts and promote the development of the smart contract industry. At the same time, formulating a unified specification for smart contracts can also facilitate experts in various fields to communicate under a unified model understood by both parties, thereby promoting the development of smart contracts.

2. Intellectual Property Transaction Smart Contract Design

2.1. Smart Contract Structure and Process. In this section, according to the characteristics of intellectual property transactions, the intellectual property smart contract structure is set up into four parts: property rights transaction, rights and responsibilities analysis, transaction arbitration, and transaction traceability (as shown in Figure 1). These four parts apply to the process of intellectual property transactions different stages.

(1) Property rights transaction: it is suitable for the initial stage of the transaction, and both parties negotiate the contract and write the contract content into the blockchain after the negotiation is completed

(2) Analysis of rights and responsibilities: it is applicable to whether there is any violation of regulations on both sides in the transaction process, and the violating party shall be held accountable and punished

(3) Transaction arbitration: applicable when the transaction ends abnormally and the defaulting party is judged to be punished, and the defaulting party may apply for arbitration if there is any objection

(4) Transaction traceability: in case of disputes after the transaction, trace the source according to the unique number of the intellectual property rights and check the historical circulation records of the property rights, so as to avoid some property rights disputes

These four parts will set different trigger response rules. When the preset conditions are met, the corresponding smart contract rules will be automatically executed. In the property right transaction stage, the structure specified by the transaction (Table 1) needs to be stored in the blockchain, where IntellectualCode is the intellectual property service number to be traded; CooperationCode is the unique code generated by the system; the initial value of Status is false; Step is the details of the cooperation steps, and its structure is shown in Table 2. Both parties can dynamically adjust the specific execution steps according to their needs, and the initial value of State is false; the remaining fields are provided by both parties of the transaction and are stored in the blockchain after the negotiation is completed. This will be used as a basis for further implementation in the subsequent process.

Based on the structure of property rights transaction, rights and responsibilities analysis, transaction arbitration, and transaction traceability, this paper designs smart contract transaction specifications on the basis of the traditional intellectual property transaction process, so as to ensure that the rights of buyers and sellers can be protected in the process of intellectual property transactions. At the same time, it can provide a reference for the design process of intellectual property-related smart contracts in the future. At present, intellectual property transactions can be mainly divided into two categories. One is direct one-time transaction, for example, trademark rights and copyrights can be directly transferred. The buyer and the seller can reach an agreement to sign a contract. The other type is service-type transactions, which take a long time to complete. The specification flow is shown in Figure 2.

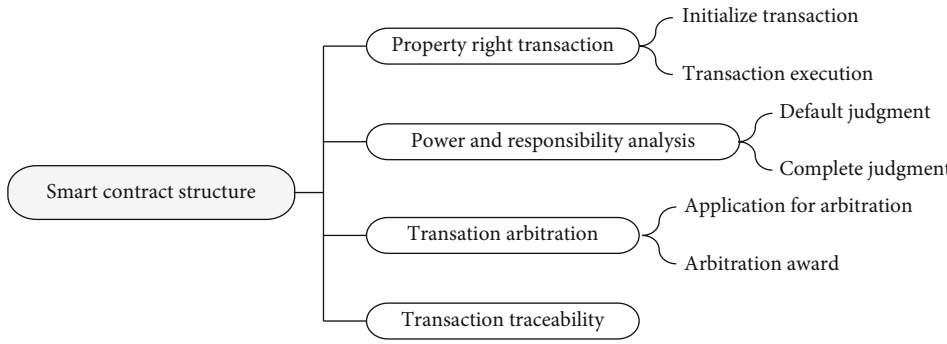


Figure 1: Intellectual property transaction contract structure.

Table 1: Transaction structure.

Field name	Type	Explanation
UserId	String	Sponsor ID
IntellectualCode	String	Intellectual property number
CooperationCode	String	Contract no
Time	Int64	Cooperation time
CompletionCycle	String	Complete cycle
Requirement	[]string	Cooperation requirements
Step	[]step	Cooperation steps
OperationTime	Int64	Operation time
Price	float64	Price
Status	Bool	Contract status
FirstSign	String	Party A's signature
SecondSign	String	Party B's signature

Table 2: Cooperation step structure.

Field name	Type	Explanation
TimeNode	int64	Time frame
CompletedItems	String	Get things done
State	Bool	Status

The following contents in this section describe in detail the specific work of property rights transaction, rights and responsibilities analysis, transaction arbitration, and property rights traceability structure.

2.2. Property Rights Transaction

2.2.1. Initialize the Transaction. Before the transaction, the service provided by the seller needs to be verified. After the verification is completed, the two parties agree to create a contract. Here, we assume that all services meet the requirements. Successful contract creation needs to satisfy $\mathbb{C}\{U_1, U_2, S, F, R, TC\}$. The meaning of $\mathbb{C}$ is that the buyer U_1 puts forward a demand S (Step) to the seller U_2, and U_2 gives a feedback result F after processing, and if S is reached, a result R is generated. $\mathbb{C}$, S, F, and R are bool types to indicate whether the demand can be achieved, and the contract signing result and TC (time constraint) is the constraint time for the contract signing.

There may be several different forms in the life cycle of a transaction contract, and the state transitions are shown in Figure 3:

(1) Not generated: S, F, and R are false, indicating that the contract has not been drawn up and the demand has not been proposed

(2) To be signed (TBS): S is true, and F and R are false, indicating that U_1 has made a request, U_2 has not yet processed it, and the agreed time has not exceeded and is waiting for U_2 to process it

(3) Executing: S and F are true, R is false, indicating that U_2 can meet the requirement S, and the contract has been formulated successfully and is being executed

(4) Complete: S, F, and R are true, indicating that U_2 can meet the demand S, and the contract is completed within the agreed time

(5) Fail: S and R are true, F is false, indicating that U_2 cannot meet the requirement S, and the contract has failed to be formulated within the agreed time

(6) Breach of contract (BOF): S is true, and F and R are false, which means that U_1 has made a request, U_2 has not yet processed it, and beyond the agreed time, it is determined that U_2 is in breach of contract, and the contract formulation has failed

2.2.2. Transaction Execution. The transaction execution is that after the two parties sign a contract, U_2 performs services according to the time node according to the requirements proposed by U_1. In the process of intellectual property transactions, the rights and obligations of the parties are formulated at a specific time. For example, in this example, we set the following: the buyer needs to deposit funds into the platform within t days of signing the contract; otherwise, it will be judged as a breach of contract. Therefore, the time benchmark is essential in the smart contract. The time benchmark model for the definition of intellectual property smart contracts is shown in Figure 4. The time benchmark corresponds to the cooperation requirements

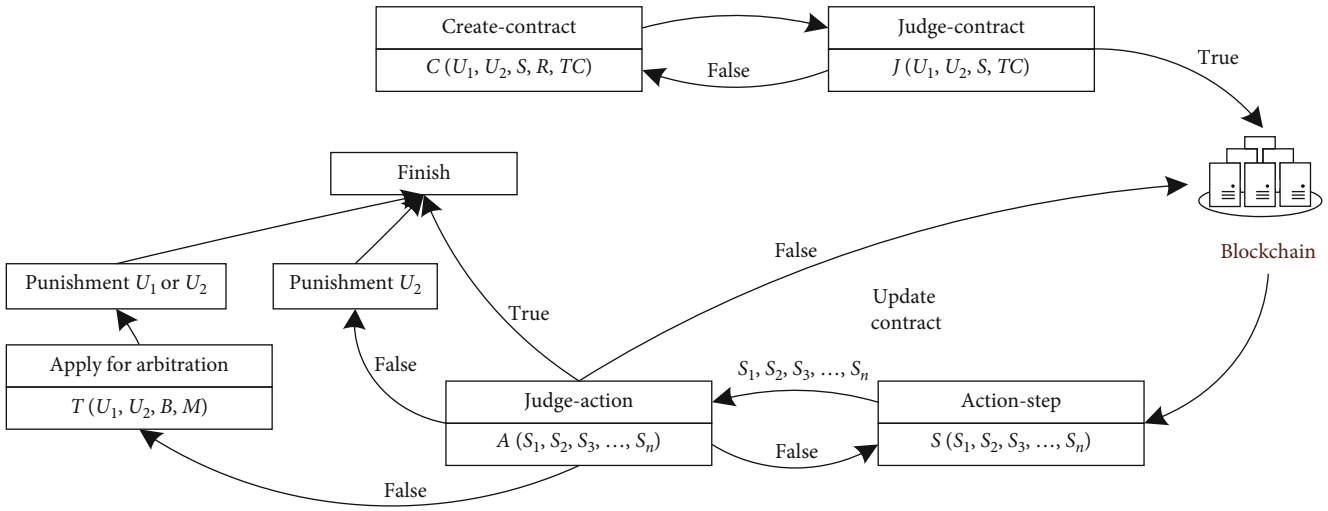


Figure 2: Execution specification of intellectual property transaction contract.

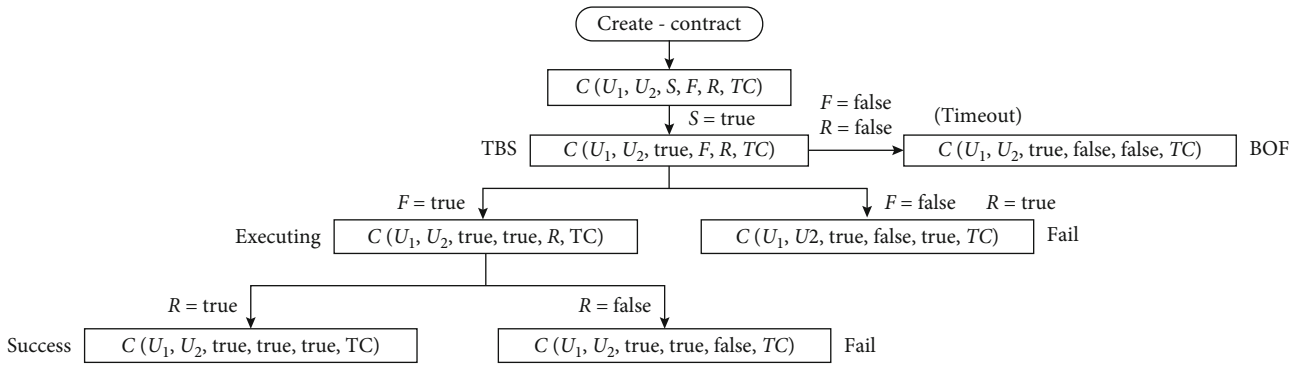


Figure 3: State transition model.

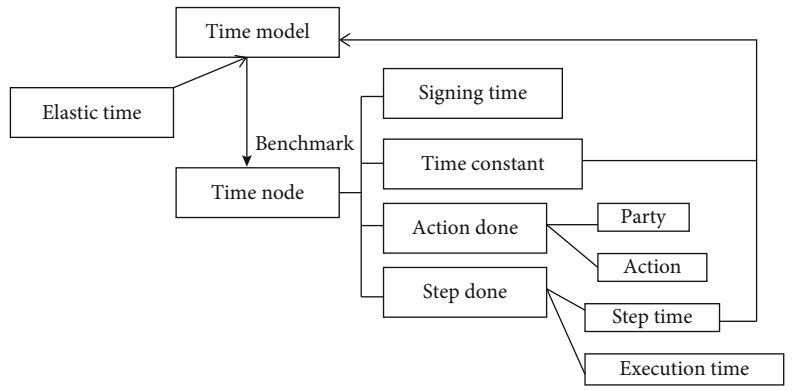


Figure 4: Time base model.

declared in the contract. Here, the step model is defined as follows:

Step 1 name: (req1, req 2, …, req *n*, constraintTime)

Step 2 name: (req1, req 2, …, req *n*, constraintTime)

The time model is used to check the relationship between the current time and the defined time node. Its submodules include StepTime and TimeConstant, which are used to check whether the relationship between the current time and the node time is earlier or later than the node time, which is defined as(before||after) benchmark. Before means that the current time is before the benchmark, and after means after the benchmark.

Time node includes four submodules, TimeConstant, SigningTime, ActionDone, and StepDone. SigningTime represents the time when the contract was signed; StepDone includes the completion time and actual execution time of the steps stipulated in the contract; TimeConstant represents a time constant and a constraint time needs to be defined in each requirement. TimeConstant can be expressed as TimeConstant = constraintTime – SigningTime. ActionDone

indicates the execution time of an action. If U_2 does not perform any operation, the function returns nil, and the time of returning nil should be earlier than the earliest time in StepDone. Here, each completed step of U_2 is defined as an action: action = stepName {party, illustrate, content, sign} where party refers to the action party, illustrate refers to a brief description, and content refers to the submitted content. The action type is bool, and the initial value is action = false. Set action = true when U_2 completes the requirements and is verified, which indicates that the action has been completed. If it fails to pass the verification, it sets action = false.

ElasticTime represents elastic time. ElasticTime is similar to TimeConstant. If U_2 fails to meet the requirements of U_1 within the constraint time, it can apply to U_1 for a delay time. After U_1 agrees, the elastic time can be increased. For example: step 1 requires that req1 should be completed within t days, and U_2 cannot complete it as scheduled during the execution period. In this case, you can apply for a delay of n days before submitting, and benchmark + n days after U_1 agrees.

2.3. Power and Responsibility Analysis. During the execution of intellectual property transactions, there will be some reasons that the transaction cannot be completed. Here, we will determine the rights and responsibilities of the transaction according to the behavior of U_1 and U_2.

After the contract is signed, U_1 needs to deposit the security deposit (buyerBond) into the system account. Assuming that the transaction is completed in n stages and the transaction price is price, then U_1 needs to have buyerBond $\geq$ price$^*(1/n)$ in the account in the initial stage: funds; otherwise, it will be judged that the contract signing has failed. U_2 also needs to deposit a margin (sellerBond) into the account to ensure that relevant services can be provided on time.

After the contract is signed and enters the transaction execution stage, the deposit will be frozen in the smart contract, and the judgment result should be the following 4 cases:

Case 1. During the execution process, U_2 submits the completion status to U_1 within the step constraint time, and U_1 reviews it. After U_1 reviews and meets the requirements, the bond frozen in the smart contract will be transferred to U_2 account, and the next step will be executed.

Case 2. During the execution process, U_2 submits the completion status to U_1 within the step constraint time, which is reviewed by U_1. If the requirements of U_1 are not met, U_1 rejects it. If the time is still within the constraint time, U_2 can submit it again until the step is complete. After this step is completed, the bond frozen in the smart contract will be transferred to the U_2 account.

Case 3. During the execution process, U_2 fails to complete the task of this stage within the constraint time and does not apply for ElasticTime or fails to pass the application. It is determined that U_2 is in breach of contract, and the bond frozen in the smart contract will be returned to the U_1 account within t days. At the same time, U_2 pays liquidated damages, and the transaction fails.

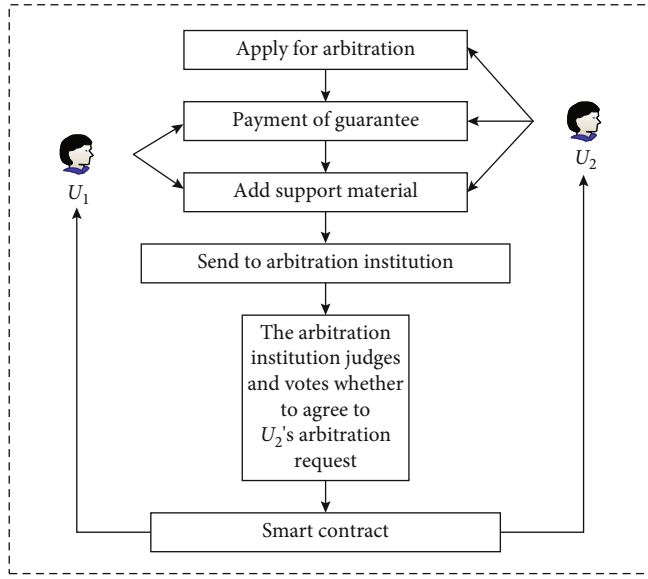


Figure 5: Transaction arbitration.

Case 4. U_1 needs not to deposit bond $\geq$ price$^*(1/n)$ in the next stage within t days after the completion of each step, determines that U_1 is in breach of contract, and pays liquidated damages to U_2 at the same time, and the transaction fails.

If U_2 believes that U_1 maliciously refuses the services it provides, resulting in failure to complete the task on time, it can apply for transaction arbitration. The arbitration process is described in the next subsection.

2.4. Transaction Arbitration. When U_2 believes that U_1 maliciously refuses the services it provides, it can apply for transaction arbitration within t days and pay n% of the transaction price as a security deposit. U_1 needs to pay the same amount of security deposit; otherwise, the bond frozen in the smart contract will be deducted. During the arbitration process, both parties U_1 and U_2 need to provide arbitration materials and then wait for the arbitration result. In order to ensure the fairness of the arbitration result, the arbitration institution shall be independent of the enterprises of both parties or the enterprises where the individuals belong, and the final arbitration result shall be voted on by the arbitration institution: ArbitrationResult = AgreeArbitration/ArbitrationNum.

ArbitrationResult indicates the arbitration result, AgreeArbitration indicates the number of arbitrations agreed, and the ArbitrationNum indicates the total number of institutions participating in the arbitration. Arbitration is deemed successful when ArbitrationResult > 2/3. The arbitration process is shown in Figure 5.

After being adjudicated by the arbitration institution, the arbitration results and handling methods are defined as the following two types:

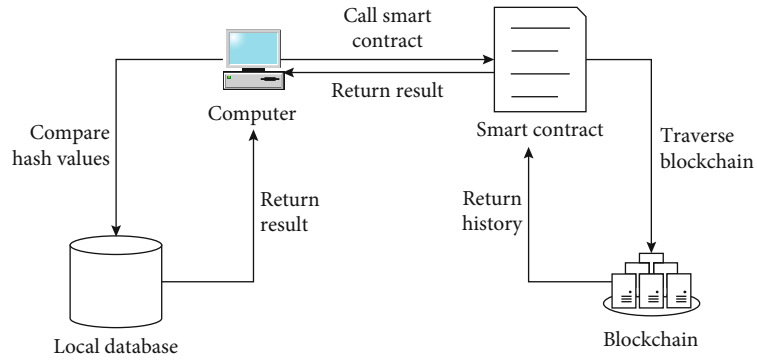


Figure 6: Transaction traceability.

```
[root@iZ2zegj5m4qenw4knztuppZ ~]# curl 127.0.0.1:40098/create -POST -d '{"hold_id":
"001","service_code":"000","Requirement":["Supports high concurrency"],"cooperation_
code":"002","cooperation_type":"resarch","time":"2022-05-06","first":"001","second":"00
2","start_date":"2022-05-06","end_date":"2022-08-06","state":"executing","status":true,
"sketch":"Completion of the IP trading platfotm within two weeks","step":[{"time_node":
"2022-05-06","completed_items":"Complete front-enddevelopment","operator":"001"}],
"operation_time":1651911194,"price":"2000.0"}'
```

(a)

```
{"data": "{\"object_type\":\"21bd0eb929c79556e66c465d65ea56b6\",\"hold_id\":\"001\",
\"service_code\":\"000\",\"Requirement\":[\"Supports high concurrency\"],\"cooperation_
code\":\"002\",\"cooperation_type\":\"resarch\",\"time\":\"2022-08-06 02:52:14\",\"first\"
:\"001\",\"second\":\"002\",\"start_date\":\"2022-05-06\",\"end_date\":\"2022-08-06\",\"st
ate\":\"executing\",\"status\":true,\"sketch\":\"Completion of the IP trading platfotm with
in two weeks\",\"step\":[{\"time_node\":\"2022-05-06\",\"completed_items\":\"Complete
front-end development\",\"operator\":\"001\",\"state\":false}],\"operation_time\":1659754
34,\"price\":\"2000.0\"}","msg": "Complete! ","txId": "06d3e164bd71b0aa2202d9e117531
5f84e5366b3e87bfb12436e2be77faadc12"}
```

(b)

Figure 7: Contract generation.

Result 1: if $ArbitrationResult > 2/3$, then the arbitration is successful, and U_2's application for arbitration is successful. The smart contract returns the arbitration security deposit to the U_2 account and transfers the security deposit of U_1 frozen in the smart contract to the U_2 account. The arbitration bond paid by U_1 will be distributed equally to the institutions participating in the arbitration

Result 2: if $ArbitrationResult < 2/3$, it means that the arbitration failed, U_2 failed to apply for arbitration, the smart contract will return the arbitration security money to the U_1 account, and the default penalty will be executed according to the original contract processing method. The arbitration bond paid by U_2 will be distributed equally among the institutions participating in the arbitration

2.5. Transaction Traceability. Intellectual property issues currently exist in a large number of industries, such as property fraud, property counterfeiting, and illegal occupation. There may be legal disputes after the property rights transaction. This model combines the nontampering characteristics of the blockchain to design the property rights traceability structure. The history structure is added to the property rights transaction structure designed in the second part, and its structure is similar to the property rights transaction structure.

Each transaction of intellectual property rights will generate different contracts, but the number of registered intellectual property rights is unique. So, when querying the traceability of intellectual property rights, use IntellectualCode as the key value to query the blockchain records and traverse each block through the key value. After traversing all blocks, the historical flow records and hash values related to IntellectualCode are returned, and the hash value of the traceability information corresponds to the local database; that is, the hash values on the chain and off the chain are compared during the query to verify the traceability, whether the data has been tampered with. The transaction traceability rules are shown in Figure 6.

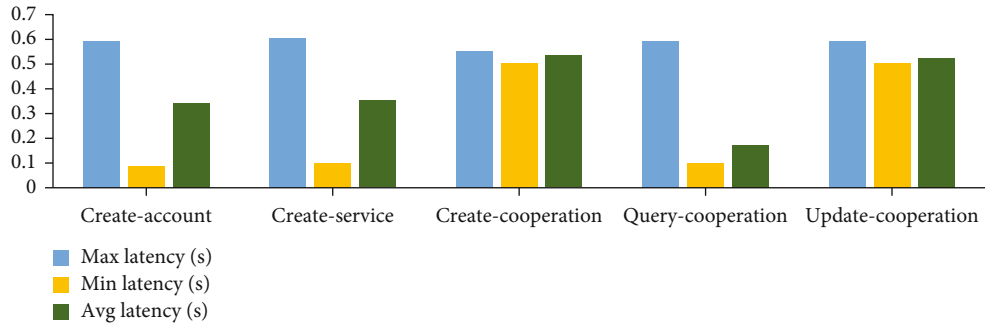
```
[root@iZ2zegj5m4qenw4knztuppZ ~]# curl 127.0.0.1:40098/check -POST -d '{"id":"001",
"skip":"0","code":"001","status":"true"}
```

(a)

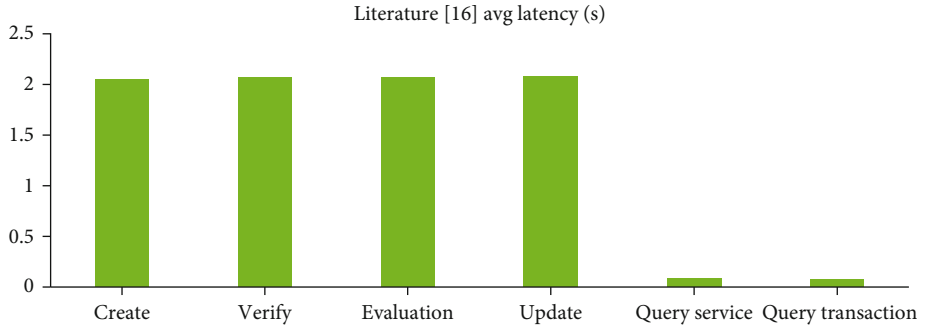

```
{"data": "{\"object_type\":\"21bd0eb929c79556e66c465d65ea56b6\",\"hold_id\":\"001\",
\"service_code\":\"000\",\"Requirement\":[\"Supports high concurrency\"],\"cooperation_
code\":\"002\",\"cooperation_type\":\"resarch\",\"time\":\"2022-08-06 02:52:14\",\"first\"
:\"001\",\"second\":\"002\",\"start_date\":\"2022-05-06\",\"end_date\":\"2022-08-06\",\"s
tate\":\"executing\",\"status\":true,\"sketch\":\"Completion of the IP trading platfotm with
in two weeks\",\"step\":[{\"time_node\":\"2022-05-06\",\"completed_items\":\"Complete f
ront-end development\",\"operator\":\"001\",\"state\":true}],\"operation_time\":165977328
,\"price\":\"2000.0\"}","msg": "Complete! ","txId": "ce642f3b804c837f47561273c70e451e
5888ee6c95a9fa05f8177524d61bbc7b"}
```


(b)

Figure 8: Stage judgment.



(a)



(b)

Figure 9: Test results.

3. System Implementation and Performance Evaluation

Based on the above description, this section writes the smart contract formulated by the intellectual property transaction contract and tests the smart contract based on the Hyperledger Fabric platform. In terms of computer overhead, we implement a prototype of the scheme to evaluate its performance. The experiments were tested on a server with CPU-8 cores running CentOS (64-bit). In Fabric v1.4.4, we built a blockchain network which consists of 1 orderer node, 2 organization nodes, and 2 peer nodes. Chaincode is code written in the GO language, which can read and write operations on the blockchain.

3.1. System Implementation. Based on the intellectual property transaction smart contract structure described in this paper, combined with the intellectual property transaction scenario, this paper implements a set of intellectual property transaction contracts to sign smart contracts and conducts test experiments based on the above experimental environment configuration. The experiment sets up two users and one intellectual property commodity to test the smart contract structure, and the entities involved in the experiment have been verified.

The process of signing the contract is shown in Figure 7, in which Figure 7(a) shows the parameters to be provided by the users of both parties to sign the contract, the peer node performs the chain operation, and the smart contract feedback result is shown in Figure 7(b).

After the contract is signed, according to the completion progress and completion quality of the provider, staged receipt or rejection can be made. After the buyer agrees to sign for receipt, the smart contract will transfer the funds to the seller's account. The staged receipt process is shown in Figure 8. Figure 8(a) shows that the buyer operates on the contract status, and Figure 8(b) shows the smart contract feedback results. Other renderings are not shown due to space reasons.

3.2. Performance Evaluation. Based on the above model, the smart contract is stress tested. The experiment mainly tests the representative contract functions in the smart contract model, such as service registration, service query, contract registration, and contract modification. We focus on the average transaction delay characteristics of smart contracts. The test results of this paper are shown in Figure 9(a), which shows the maximum, minimum, and average delay times of the tested functions. It can be seen from the figure that the model proposed in this paper has an average delay of 0.4 s in the creation of contracts, services, and update functions and an average delay of 0.17 s in the query function. At the same time, it is compared with the solution of the literature [16]. The test results of the solution proposed in the literature [16] are shown in Figure 9(b). The average delay of the creation, verification, and update functions is 2 s, and the average delay of the query function is 0.08 s.

Combining the above experiments, it can be seen that the query delay is generally smaller than the creation delay. This is because more logical judgments is required in the creation of services and contracts, to ensure the rights and interests of both buyers and sellers. At the same time, consensus nodes are required to generate blocks by consensus, while the query service does not. Operations such as consensus are required; so, the latency is low. Compared with literature [16], it can be concluded that the proposed scheme can save time in creating functions, and the query function delay is slightly higher than that of literature [16]. The overall test shows that the model meets the actual production requirements, and the transaction delay is better than the existing model structure, and the user experience can be guaranteed.

4. Conclusion and Outlook

The smart contract model based on the intellectual property transaction scenario designed in this paper is divided into four parts: property rights transaction, rights and responsibilities analysis, transaction arbitration, and transaction traceability. It realizes the formal definition of smart contract structure and provides intellectual property rights. The unified and standardized transaction process is convenient for scholars in various fields to conduct discussions based on this specification and promote the development of smart contracts. The model has been deployed and applied on the Beibu Gulf comprehensive technology service platform, verified the design ideas and related technologies of the model, and achieved good application effect. It provides a good reference for solving the problems of low sharing level and high development cost of smart contract.

In the future work, it is necessary to develop a visual smart contract modeling tool for intellectual property transactions, which can be extended in combination with the actual situation to achieve more complex functions. At present, blockchain technology also has many problems, such as long verification time and high energy consumption. In the next research center, we should also focus on improving the execution speed and throughput of blockchain. Drawing on knowledge from other directions such as artificial intelligence and big data makes smart contracts more dynamic and adaptive.

Data Availability

The data used to support the finding of this study are included within the article.

Conflicts of Interest

The authors declare no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Project under Grant 2018YFB1404400, Major Science and Technology Project of Haikou under Grant 2020-009, and Key R&D Project of Hainan Province under Grant ZDYF2021SHFZ243.

References

[1] Q. Lu and X. Xu, "Adaptable blockchain-based systems: a case study for product traceability," *IEEE Software*, vol. 34, no. 6, pp. 21–27, 2017.

[2] N. Szabo, "Formalizing and securing relationships on public networks [J]," *First Monday*, vol. 2, no. 9, 1997.

[3] "Ethereum project," [Online]. Available: https://www.ethereum.org.

[4] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: the blockchain model of cryptography and privacy-preserving smart contracts," Proc. of IEEE SP 2016, San Jose, CA, United states, 2016.

[5] A. Ford, "IBM to create blockchain for cobalt; program will trace suppiy line of cobalt from Congo [J]," *Mining Engineering*, vol. 71, no. 3, p. 12, 2019.

[6] S. Lin, Z. Lei, and L. Desheng, "A review of application research based on blockchain smart contracts [J]," *Application Research of Computers*, vol. 38, no. 9, pp. 2570–2581, 2021.

[7] A. Singh, R. M. Parizi, Q. Zhang, K. K. R. Choo, and A. Dehghantanha, "Blockchain smart contracts formalization: approaches and challenges to address vulnerabilities," *Computers & Security*, vol. 88, pp. 101654–101654. 16, 2020.

[8] X. He, B. Qin, Y. Zhu, X. Chen, and Y. Liu, "SPESC: a specification language for smart contracts [C]," in *IEEE Computer Software & Applications Conference*, IEEE, 2018.

[9] O. Choudhury, N. Rudolph, I. Sylla, N. Fairoza, and A. Das, "Auto-generation of smart contracts from domain-specific ontologies and semantic rules," IEEE, 2019.

[10] T. Astigarraga, X. Chen, Y. Chen et al., "Empowering business-level blockchain users with a rules framework for smart contracts [c]," in *The 16th International Conference on Service-Oriented Computing (ICSOC)*, Springer, Cham, 2018.

[11] Y. Zhu, W. Song, D. Wang, D. Ma, and W. C. Chu, "TA-SPESC: toward asset-driven smart contract language supporting ownership transaction and rule-based generation on blockchain [J]," vol. 99, IEEE Transactions on Reliability, 2021.

[12] P. W. Wang, H. T. Yang, J. Meng, J. C. Chen, and X. Y. Du, "Formal definition for classical smart contracts and a refernce implementation," *Ruan Jian Xue Bao/Journal of Software*, vol. 30, no. 9, pp. 2608–2619, 2019, (in Chinese).

[13] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, "Evaluation of logic-based smart contracts for blockchain systems [c]//ruleml," Springer, Cham, 2016.

[14] S. Alqahtani, X. He, R. Gamble, and P. Mauricio, "Formal verification of functional requirements for smart contract compositions in supply chain management systems [C]," *Hawaii International Conference on System Sciences*, 2020.

[15] Z. Meng, T. Morizumi, S. Miyata, and H. Kinoshita, *Design Scheme of Copyright Management System Based on Digital Watermarking and Blockchain [C]//2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE Computer Society, 2018.

[16] B. Zhang, W. Feng, M. Huang, S. Feng, and X. Zheng, *Smart contract model of sci-tech services transaction based on consortium blockchain*, 2021 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS), 2021.