electronics

Article

Research on the Architecture of Transactional Smart Contracts Based on Blockchains

Zhixiang Liu [1], Wenlong Feng [1,*], Yu Zhang [2] and Chengcheng Zhu [1]

[1] School of Information and Communication Engineering, Hainan University, Haikou 570228, China; 21220854000159@hainanu.edu.cn (Z.L.); 21220854000205@hainanu.edu.cn (C.Z.)
[2] School of Computer Science and Technology, Hainan University, Haikou 570228, China; yuzhang2015@hainanu.edu.cn
* Correspondence: fwlfwl@163.com

Abstract: At present, smart contracts are designed based on the application field, and their structure and functions are closely related to specific businesses. Even smart contracts within the same field and the same business have different structures and functions due to different developers, resulting in structural confusion, repeated development, and low levels of sharing. In response to this problem, this study conducts a full investigation of smart contracts in various fields, using big data technology to compare and analyze the structures of each contract and extracting the common content of their main bodies to study each feature, as well as to conduct induction and fusion. This study also generally designs a hierarchical structure and formulates structural modules such as transaction rules and the analysis of rights and responsibilities, as well as a reward and punishment mechanism. Data traceability is established, and the overall architectural specification of smart contracts is constructed. Additions, deletions, and improvements are made based on specific application environments to realize the dynamic updates of the architecture of contracts. Experiments show that the architecture of contracts can realize the various functions required in a transaction, solve the problem of the repeated development of current transactional smart contracts, and improve the sharing level.

Keywords: blockchain; smart contract; transaction; architecture

Citation: Liu, Z.; Feng, W.; Zhang, Y.; Zhu, C. Research on the Architecture of Transactional Smart Contracts Based on Blockchains. *Electronics* **2023**, *12*, 3923. https://doi.org/10.3390/electronics12183923

Academic Editor: Rameez Asif

Received: 5 August 2023
Revised: 5 September 2023
Accepted: 14 September 2023
Published: 18 September 2023

1. Introduction

Smart contracts are computer programs that automatically execute contracts based on blockchain technology. They are designed to enforce and verify the fulfillment of contractual terms in a decentralized, transparent, secure, and reliable manner. The concept of smart contracts was first proposed by Szabo [1] in 1995 and has been widely used and developed in blockchain platforms such as Ethereum. With the rapid development of the blockchain and the widespread popularity of the Internet, the research and application of smart contract technology has also attracted the attention of many scholars [2]. As a new form of contract, smart contracts write a contract's terms into a computer programming language and start the automatic execution of the contract through the realization of preset conditions. When the conditions are met, the contract takes effect. It has the characteristics of decentralization and trustworthiness. It is also of a high quality, is not easy to be tampered with, and so on. Therefore, the use of blockchain smart contract technology can effectively solve the current problems of relying on third parties in various transactions, and contracts being easily interfered with by external factors such as civil disputes, to ensure the authenticity and reliability of information, as well as to prevent transaction fraud and false transactions. Transactions realize the definition of rights and responsibilities, and safeguard the legal and legitimate rights of both parties to the transaction.

At present, there are many application platforms for smart contracts. Ethereum is an open-source and general-purpose public blockchain platform with smart contract functions. Through its dedicated cryptocurrency, Ether, the Ethereum virtual machine is used to process point-to-point contracts [3]. Hyperledger Fabric is a modular, open-source, enterprise-level, and permissioned distributed ledger technology platform that is designed to be used in an enterprise environment, providing functions such as pluggable implementations that support different components and channel creation [4]. The enterprise operation system (EOS) is a blockchain underlying public-chain operating system designed for commercially distributed applications, with performance extensions for traditional applications [5].

Cai [6] provided a method and device for processing rule changes in smart contracts. The method provided can integrate a business rule engine in a smart contract, so that business personnel can formulate business rules conveniently and simply. Wang [7] analyzed and compared 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. Hu [8] pointed out that a formal verification framework applied to the life cycle of smart contracts should be proposed. The Prome modeling language was used to model smart shopping contracts, and SPIM was used to model monitoring to verify the effect of formalization on smart contracts. Mavridou [9] designed smart contracts based on finite state machines, and provided a tool to create FSM on an easy-to-use graphical interface and automatically generate Ethereum contracts. Developers can easily add this to their contracts for enhanced security and functionality. Bai [10] provided a general and formal description of the smart contract template. The model-checking tool SPIN was used to verify the correctness and necessary properties of the smart contract template, and verify the nature of smart contracts. Xu [11] proposed a digital bond transaction contract system based on blockchain technology and studied the system's architecture using blockchain technology to optimize the structure of the bond trading system, as well as using blockchain technology to develop the software's architecture. Through a performance test of the system, the advantages of the system's architecture in practical applications were verified. Han [12] proposed a general framework for a blockchain platform to enable peer-to-peer (P2P) energy transactions in retail electricity markets. It focuses on finding energy-matching pairs from the supply and demand side, encourages direct energy transactions between producers and consumers, and realizes a complete energy transaction process. Shi [13] proposed a transaction model based on legal contracts using smart contract templates. This model can dynamically construct, store, and call smart contracts based on smart contract templates and transactional contract contexts, thereby improving the reusability of smart contracts and reducing their difficulty of use. Nazari [14] proposed a safe and automated blockchain-based framework, using the Solidity programming language and the MetaMask wallet to create smart contracts, allowing energy producers and consumers to conduct energy trades without intermediary entity interaction, and enhancing security and privacy.

To sum up, smart contracts have been widely used in transactions, but all smart contracts are designed based on the application field, and the structure and function of each smart contract is different, so the structure is chaotic, the development is repeated, and the level of sharing is low; there is not yet a standardized definition of the structure of smart contracts. Therefore, this article aims to tackle the current problems of smart contracts for transactions and establishes the general architectural specification of smart contracts, aiming to provide a general blueprint for reference and use when writing smart contracts for transactions. This architecture covers the key aspects and necessary components in the process of writing smart contracts, from transaction rule setting to breach of contract liability, from power and responsibility analysis to the reward and punishment mechanism, as well as data traceability, etc., to comprehensively define the design points of smart contracts for transactions. At the same time, the layered structure of the contract enables the targeted expansion of certain layers without affecting the performance of other layers, and the scalability of the contract can be realized.

2. Contract Structure Design

To fully understand the situation of smart contracts in the trading field, one can conduct in-depth research and data collection on the architecture of smart contracts. First, consult academic papers, research reports, and other literature sources to obtain design cases and related materials on smart contracts. The literature covers an in-depth analysis of smart contract design patterns, as well as contract structures and functional characteristics. In addition to extensively reviewing comprehensive information on smart contracts, it focuses on in-depth research on smart contract applications in specific industries. For example, focus on specific scenarios such as intellectual property transaction contracts and logistics tracking contracts in the supply chain field to obtain industry-specific design patterns and experiences. Focused research can provide specific examples of the application of smart contracts in specific fields, and can provide a more comprehensive understanding of the application and effect of smart contracts in actual business. Through case analysis and cross-industry comparison, it is possible to gain a comprehensive understanding of the design and application of the architecture of smart contracts, providing valuable references and guidance for future smart contract development. The research results are shown in Table 1.

Table 1. Research on smart contracts' architecture.

Field	Contract Structure
finance	1. Transaction rules: includes asset transfer rules, buying and selling rules, transaction fee rules, and other rules; 2. Power and responsibility analysis: clarifies the roles and authorities of all parties involved in the financial transaction, and determine the powers and responsibilities for when a transaction fails; 3. Compliance and regulation: includes the consideration of compliance audits and regulatory requirements; 4. Reward and punishment mechanism: sets incentive measures to encourage the smooth completion and active participation of transactions; at the same time, it stipulates punishment measures for violations to ensure the integrity and fairness of transactions; 5. Cross-chain interaction: in the financial field, asset transfers and transactions between different blockchains may be involved; 6. Data traceability: records relevant data and the history of financial transactions.
supply chain	1. Transaction rules: including logistics tracking, product distribution, supply chain payment, etc.; 2. Power and responsibility analysis: participants include suppliers, logistics companies, retailers, etc., and multiple people need to be responsible for the supply chain; 3. Logistics tracking: provides more effective logistics management and scheduling; 4. Product quality traceability: records product quality information and quality inspection results, and realizes product quality traceability and monitoring; 5. Security and privacy protection: supply chain transactions involve multiple links and participants, and security and privacy protection need to be considered; 6. Reward and punishment mechanism: the performance evaluation and reward and punishment mechanism of each link in the supply chain motivate the enthusiasm of participants.
intellectual property	1. Trading rules: including intellectual property verification rules, copyright transfer rules, copyright licensing rules, etc.; 2. Analysis of rights and responsibilities: for clearly understanding the responsibilities of copyright owners, copyright users, and platform parties; 3. Data encryption: data encryption is used to protect the security and privacy of transactions involving intellectual property; 4. Transaction traceability: records the use and transfer of intellectual property rights, and realizes the monitoring and tracking of intellectual property rights; 5. Punishment mechanism: sets punishment measures for violations in transactions, and warns all parties to abide by transaction rules and copyright agreements.

Table 1. *Cont.*

Field	Contract Structure
real estate	1. Transaction rules: real estate verification rules, transaction confirmation rules, information recording rules, etc.; 2. Rights and responsibilities analysis: the rights and obligations of both parties to the transaction, including payment, transfer procedures, etc., and the ownership of responsibilities in real estate transactions; 3. Property information records: record and manage real estate property information, including property costs, maintenance records, etc.; 4. Real estate evaluation and pricing: to ensure fair and reasonable transactions; 5. Punishment measures: include additional handling fees, transaction cancellation, etc.; 6. Data traceability: records relevant data and the history of real estate transactions, including payment records during the transaction process, asset handover information, etc.

After conducting research on the architecture of smart contracts for transactions in various fields, analyze and compare these architectures to extract the common content of their main bodies, summarize, design a hierarchical structure on the whole, and build an overall specification for the architecture of smart contracts. The smart contract structure is divided into four parts: transaction rules, power and responsibility analysis, reward and punishment mechanism, and data traceability. The architecture of smart contracts is shown in Figure 1.

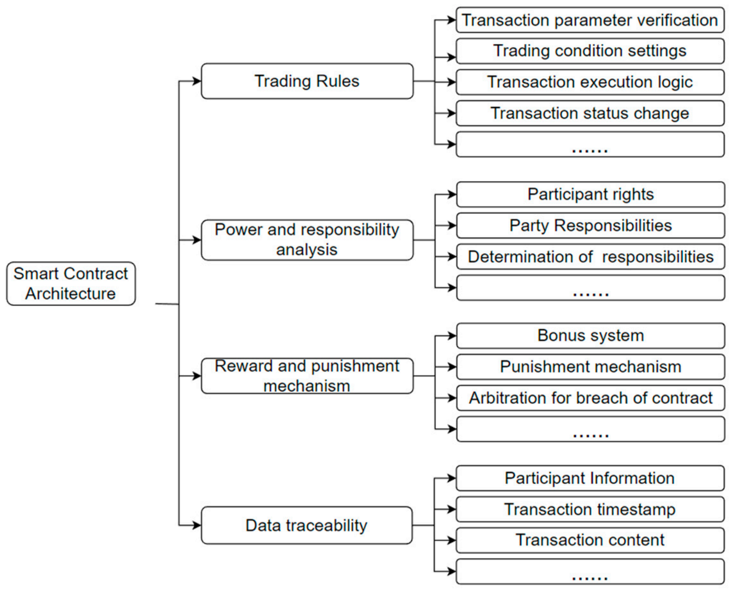

Smart Contract Architecture
Trading Rules
Transaction parameter verification
Trading condition settings
Transaction execution logic
Transaction status change
......
Power and responsibility analysis
Participant rights
Party Responsibilities
Determination of responsibilities
......
Reward and punishment mechanism
Bonus system
Punishment mechanism
Arbitration for breach of contract
......
Data traceability
Participant Information
Transaction timestamp
Transaction content
......

Figure 1. Smart contracts' architecture.

Transaction rules refer to the rules and conditions defined in a transaction's smart contract, which are used to guide and constrain the execution of transactions. By clearly defining transaction rules, smart contracts can ensure that only transactions that meet the specified conditions can be executed, avoiding illegal or incorrect transactions.

The rights and responsibilities analysis refers to clarifying the rights and responsibilities of each participant in a transaction's smart contract. This includes the rights that the creators, participants, and possible external users of a contract have in the transaction, as well as the responsibilities and obligations they need to assume in the transaction. The rights and responsibilities analysis not only involves defining the rights and obligations of both parties in a transaction, but also needs to determine the rights and responsibilities in the case of transaction failure or breaches of contract to determine which party will accept the corresponding punishment. Through the analysis of rights and responsibilities, smart contracts can ensure that the roles and rights of participants in transactions are clear, thereby reducing transaction risks and ensuring the credibility and reliability of transactions.

The reward and punishment mechanism is a mechanism to motivate contract participants to abide by a contract's rules, perform their duties, and contribute to the stability and security of the network. The reward mechanism encourages participants to actively participate in contract execution, while the penalty mechanism punishes participants who violate the contract to ensure contract execution and participants' compliance with the rules.

Data traceability is the storage and traceability of data in a contract. Smart contracts usually need to store a transaction's data and status changes for future audits and traceability. Data traceability ensures the security, reliability, and accessibility of contract data.

These four parts are the parts that need to be considered when designing smart contracts. These four parts are interrelated and together form a complete smart contract architecture to ensure the correct execution, security, and scalability of the smart contracts of transactions. Such a hierarchical design divides the intelligence into four parts, so that each part can be considered and designed independently, and the parts are interrelated to form a complete architecture of smart contracts. When designing a smart contract, it is necessary to comprehensively consider the characteristics of the transaction and fully consider the design and implementation of each part to meet the needs and goals of the transaction. At the same time, it is also necessary to follow the characteristics and limitations of the blockchain platform to ensure the efficient execution and stable operation of smart contracts. The smart contract flow chart is shown in Figure 2.

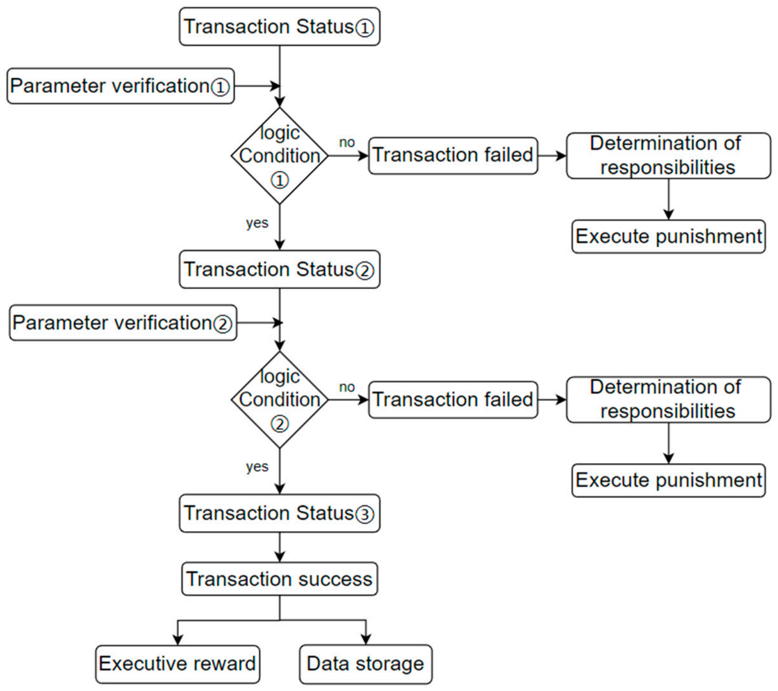


Figure 2. Contract flow chart.

2.1. *Trading Rules*

Transaction rules are the basis and guarantee of smart contracts, which define the operating logic and execution conditions of a contract. Reasonably designed transaction rules can ensure the safety, reliability, compliance, and fair execution of contracts, to achieve the expected goals and effects of smart contracts.

2.1.1. Transaction Parameter Verification

Transaction rules should define the validation requirements of transaction parameters. This includes verifying that the parameters provided by a transaction's participants conform to the expected format, type, and value range. Through a comprehensive verification of a transaction's parameters, smart contracts can prevent potential security breaches and malicious behaviors, ensuring the safety, reliability, and compliance of transactions. The verification of these parameters is a very important part of smart contracts to ensure the correctness of a transaction's execution and the trust of its participants.

1. Time parameters: including transaction timestamp, transaction validity period, time interval, etc., usually string-type;
2. Numerical parameters: including the amount, quantity, price, and other numerical information involved in the transaction, usually int64- or float64-type;
3. Address parameters: including contract address, user wallet address, partner address, etc., usually string-type;
4. Boolean parameter: a logical value representing true or false, used to trigger a specific condition or flag, usually bool-type;
5. String parameters: including transaction description, text information, name, etc., usually string-type.

2.1.2. Trading Condition Setting

Transaction conditions mean that when certain preset conditions are met, the contract will execute a specific logic.

1. Time limit: it is stipulated that the transaction is only valid within a specific time range;
2. Authorization verification: to check whether a transaction initiator has the authority to execute a contract; for example, it must pass a specific signature verification;
3. Status Judgment: to judge whether the execution conditions are met according to the contract status before a transaction; the next transaction process can only be executed in a specific state.

2.1.3. Transaction Execution Logic

A transaction's execution logic is the core of its smart contract, which defines the specific execution process and processing logic of the transaction. According to a transaction's rules and conditions, the transaction execution logic determines how a contract should process a transaction, including updating its status, and performing operations and possible outcomes.

1. Contract operation: according to the transaction type and parameters, this executes the corresponding contract operation, such as to transfer or store data, trigger other contracts, etc.;
2. Data update: updates the relevant data and status involved in a transaction to ensure the correct execution and results of the transaction;
3. Exception handling: handles abnormal situations that may occur during transaction execution, such as insufficient account balance, transaction failure, etc.

2.1.4. Transaction Status Transition

A transaction's status transition element is responsible for managing transaction status changes and state transitions. Smart contracts go through different states during execution, which may trigger other transaction conditions or cause changes in the status of a contract.

1. Status management: tracks the status changes during a transaction's execution process to ensure that contracts are converted using the specified conditions in different states;
2. State transition conditions: define the transition conditions between different states to ensure that a contract can correctly perform state transitions when certain conditions are met;
3. State change event: when the state of a contract changes, corresponding events or operations are triggered, such as triggering other contracts, notifying transaction participants, etc.

2.2. Rights and Responsibilities Analysis

The rights and responsibilities analysis is the basis of smart contracts, which clarifies the rights and obligations of contract participants, constrains their behavior, and ensures the safety, reliability, compliance, and fair execution of contracts. A reasonably designed rights and responsibilities analysis will help toward achieving the expected goals and effects of smart contracts, and improve the satisfaction and trust of contract participants.

2.2.1. Rights of Participants

It is necessary to clarify the rights that each participant has in a transaction. This may include initiating transactions, viewing transaction information, accessing the status of contracts, etc., to ensure that participants enjoy the rights they deserve in a transaction to meet their transaction needs.

2.2.2. Responsibilities of the Participants

It is important to determine the responsibilities and obligations of each party involved in a transaction. This may involve conditions and restrictions of transactions, as well as a duty to abide by the rules of a contract. Participants need to clearly understand their responsibilities in a transaction to ensure the normal execution of its contract and the smooth progress of the transaction.

2.2.3. Determination of Responsibilities

When a transaction fails or a breach of contract occurs, a rights and responsibilities analysis can evaluate and judge the behavior of a contract's participants to determine the responsible party and implement the corresponding punishment measures according to the contract. The identification of rights and responsibilities ensures that, in the event of transaction disputes or misconduct, responsibilities can be clarified and the legality, fairness, and stability of transactions can be maintained.

2.3. Reward and Punishment Mechanism

The reward and punishment mechanism is an incentive measure designed to encourage a contract's participants to abide by the contract's rules and actively perform the contract's tasks. A properly designed reward and punishment mechanism can increase the enthusiasm and willingness of participants to co-operate, thereby improving the execution efficiency of smart contracts, the stability of the network, and the overall development effect.

2.3.1. Credit Score System

Smart contracts can adopt a reputation scoring system to set up a reputation scoring mechanism for participants. When participants complete transactions or abide by contract rules, they can obtain certain reputation points. Depending on the level of reputation points, some rights and privileges of participants in a contract can be affected. Participants with high reputation scores may receive more rewards and greater authority, while participants with low reputation scores may face certain restrictions or penalties.

2.3.2. Fine

Smart contracts can set up a penalty mechanism, and when participants violate contract rules or fail to perform their obligations, the contract can fine them. The cost of the penalty can be set according to the degree of the breach of contract and the degree of loss, and is used to make up for the loss of the injured party.

2.3.3. Penalty Period

A contract can stipulate the processing period for breaches of contract; that is, the breaching party needs to remedy or solve the problem within a certain period. If a breach is not resolved within the stipulated period, the contract can proceed with other penalties.

2.3.4. Arbitration

The application for arbitration is usually a dispute resolution mechanism that is used when a dispute or breach of contract occurs during the execution of a contract. When the parties involved in a contract cannot solve the problem by themselves or reach an agreement, they can apply to the arbitration institution. The arbitrator reviews the terms of the contract, its performance, and the relevant evidence before making an award. The outcome of this arbitration process is enforced via smart contracts to resolve disputes.

2.4. Data Traceability

By tracing a transaction's history, the detailed information of the transaction, the identity and behavior of the participating parties, and related time stamps can be viewed. This evidence can be used to resolve disputes, adjudicate disputes, and support the arbitration process. At the same time, it provides a comprehensive visibility and transparency of a transaction's history. Anyone can trace and verify the source, participants, time, and related information of a specific transaction, which enhances the credibility of the transaction and reduces the possibility of fraud and improper behavior, including sexual misconduct.

Smart contracts record the details of transactions and store them in the blockchain. This information includes transaction time, the identity information of a transaction's participants, relevant data on items, etc. Such records ensure the traceability of transactions, enabling parties to query and verify the authenticity and integrity of transactions. When querying a certain transaction, one can query through the unique hash value of a transaction, traverse the entire blockchain, and return the transaction record with the same hash value. Each block contains the hash value of the previous block, forming an immutable chain structure. If someone tries to tamper with the previous transaction records, the integrity of the entire blockchain will be destroyed, ensuring that the transaction's information cannot be tampered with. The transaction traceability process is shown in Figure 3.

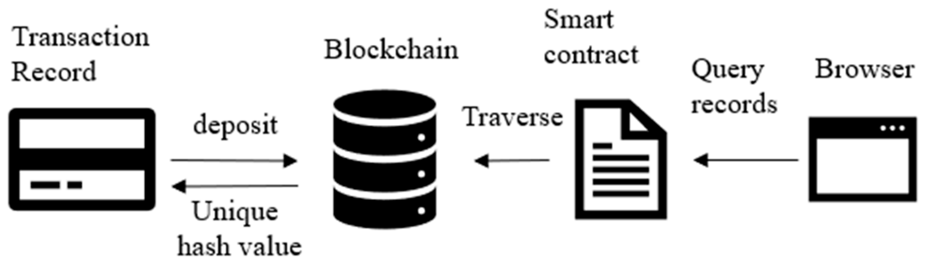


Figure 3. Transaction traceability.

3. Experiment and Analysis

3.1. Architecture Implementation

To verify the feasibility of the architecture of contracts, this experiment is based on the Hyperledger Fabric platform, and the operating system Ubuntu20.04 was selected for testing. The configuration of the virtual machine is as follows: Processor AMD Ryzen 7 5800H with Radeon Graphics, memory 4 GB, hard disk 60 GB. In this experiment, a blockchain network composed of 1 ordering node, 3 organization nodes, and 2 peer nodes was set up. The chain code part of the smart contract was developed in the Go language.

During the experiment, a commodity trading contract was designed based on the architecture of smart contracts, and the structure and rules of the contract were designed. The transaction structure (Table 2) specified the structural parameters involved in the transaction, and the parameters were defined and explained. The rules defined the trigger conditions and response rules at different stages of the transaction process. When the information received by the contract met the departure conditions, it was automatically executed.

Table 2. Transaction structure parameters.

Field Name	Type	Description
BuyerID	String	buyer ID
CommodityInf	String	commodity information
SellerID	String	seller ID
TId	String	transaction ID
Price	Float64	transaction price
Signature	String	user signature
Time	Int64	transaction generation time
ValidTime	Int64	transaction valid time
Status	String	contract status

3.1.1. The Seller's Product Is on the Shelf

In the process of a transaction, a seller first needs to put an item on the shelf and promise to carry out a commodity transaction. Firstly, the seller needs to send a commitment to commit an on-the-shelf (SPO) product to a smart contract for trading, including the seller's information, the product's information, the price, and a signature, to ensure that the seller is responsible for the product. The structure of the SPO transaction is as follows:

SellerID | | CommodityInf | | Time | | ValidTime | | Price | | Signature

This includes detailed information about the product and the guarantee of the product. If the transaction fails due to the seller, the seller will be punished. The displayed commodity information is shown in Figure 4. The product information is created by the administrator after review, and then the seller sets the price. When the guarantee status is true, it indicates that the product can be traded.

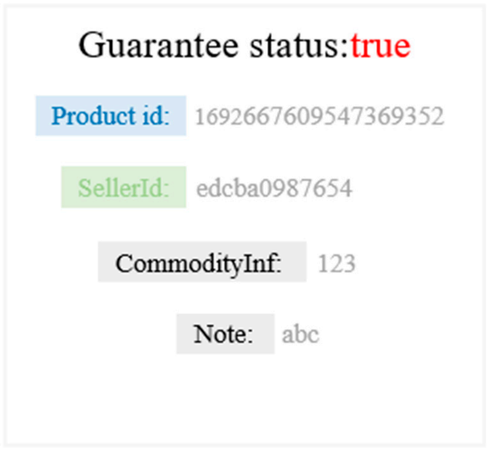

Guarantee status:true
Product id: 1692667609547369352
SellerId: edcba0987654
CommodityInf: 123
Note: abc

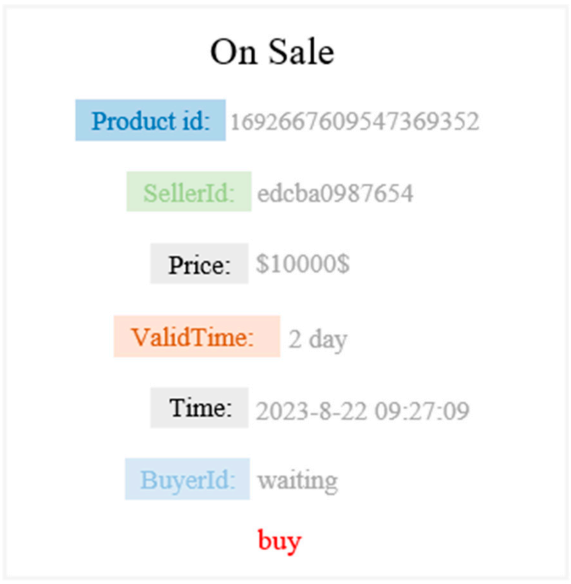
Figure 4. Commodity information.

3.1.2. The Buyer Purchases the Product

In this step, the buyer browses the product list and selects the product to be purchased. After confirmation, the smart contract is used to complete the payment operation. This step is for the buyer to pay the seller the cost of the product (BPP), and the buyer performs this action and pays the consumption amount to the smart contract account. The structure of the BPP transaction is as follows:

BuyerID | | TID | | Time | | ValidTime | | Price | | Signature

During the validity period, the buyer needs to pay the purchase funds to the smart contract, and the seller also needs to reply to the buyer's transaction within the contract's validity period. If one party fails to perform their corresponding operation within the validity period, the transaction will fail and a transaction penalty will be imposed. The user purchase interface is shown in Figure 5. After the buyer purchases, the transaction status changes to complete, and then the buyer's information is recorded and both parties perform their subsequent operations.



On Sale

Product id: 1692667609547369352

SellerId: edcba0987654

Price: 10000

ValidTime: 2 day

Time: 2023-8-22 09:27:09

BuyerId: waiting

buy

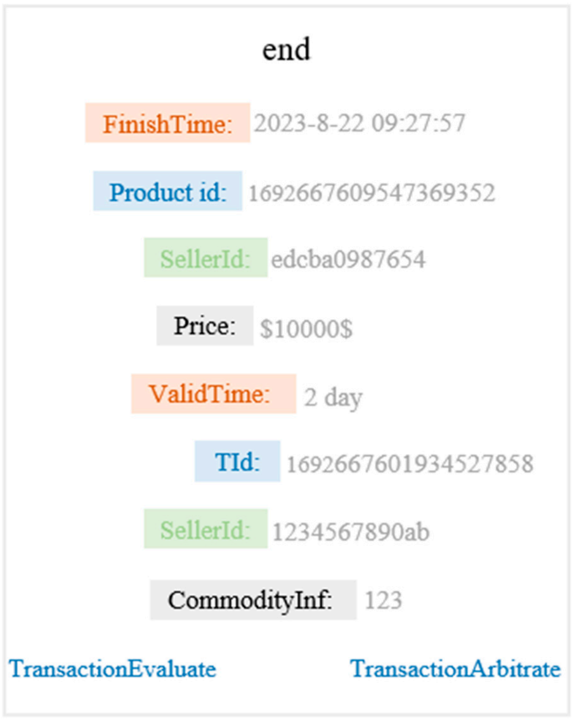
Figure 5. Buyer purchase.

3.1.3. The Seller Collects the Payment and Delivers the Product

The seller responds to the buyer's transaction request, confirms the transaction, and ships the product. After executing the BPP transaction, the seller needs to perform the final operation (SCD). The structure of the SCD transaction is as follows:

SellerID | | TID | | Time | | ValidTime | | Price | | Signature

After the smart contract receives the request from the seller and meets the conditions for releasing the transaction funds, it will transfer the fee paid from the buyer to the seller, and then the seller will deliver the product, complete the transaction, and implement incentive measures for both parties. At the same time, a transaction evaluation can be conducted. If both parties are not satisfied with the transaction, they can apply for arbitration. The transaction information is shown in Figure 6 below.



end

FinishTime: 2023-8-22 09:27:57

Product id: 1692667609547369352

SellerId: edcba0987654

Price: 10000

ValidTime: 2 day

TId: 1692667601934527858

SellerId: 1234567890ab

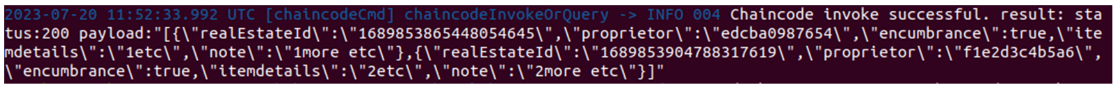
CommodityInf: 123

TransactionEvaluate TransactionArbitrate

Figure 6. Transaction information.

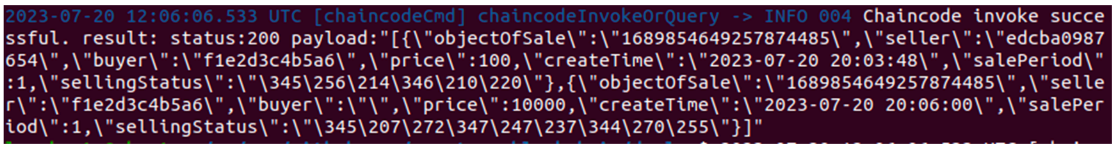
3.2. Experimental Analysis

In terms of testing the functionality of the contract, the buyer calls the contract to query the detailed information on the item. The contract's feedback result is shown in Figure 7. From the contract feedback result, it can be known that there are currently two items that have completed their registration in the system, and they are still unregistered. The status of the transaction and buyer columns are empty, and you can view detailed information such as the seller and the guarantee status of the item. Encumbrance means that the item has passed the seller's guarantee and can be traded. After the transaction is over, regardless of arbitration, the storage of transaction records is very important. Figure 8 shows the storage results of the transaction records.



```
2023-07-20 11:52:33.992 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 004 Chaincode invoke successful. result: sta
tus:200 payload:"[{\"realEstateId\":\"1689853865448054645\",\"proprietor\":\"edcba0987654\",\"encumbrance\":true,\"ite
mdetails\":\"1etc\",\"note\":\"1more etc\"},{\"realEstateId\":\"1689853904788317619\",\"proprietor\":\"f1e2d3c4b5a6\",
\"encumbrance\":true,\"itemdetails\":\"2etc\",\"note\":\"2more etc\"}]"
```

Figure 7. List of items.



```
2023-07-20 12:06:06.533 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 004 Chaincode invoke succe
ssful. result: status:200 payload:"[{\"objectOfSale\":\"1689854649257874485\",\"seller\":\"edcba0987
654\",\"buyer\":\"f1e2d3c4b5a6\",\"price\":100,\"createTime\":\"2023-07-20 20:03:48\",\"salePeriod\"
:1,\"sellingStatus\":\"\345\256\214\346\210\220\"},{\"objectOfSale\":\"1689854649257874485\",\"selle
r\":\"f1e2d3c4b5a6\",\"buyer\":\"\",\"price\":10000,\"createTime\":\"2023-07-20 20:06:00\",\"salePer
iod\":1,\"sellingStatus\":\"\345\207\272\347\247\237\344\270\255\"}]"
```

Figure 8. Transaction records.

Based on the above smart contract's architecture, the relatively important functions in the written contract were tested, including querying users, registering items, creating transactions, executing transactions, querying transaction records, and updating functions. These functions were specifically representative of the contract, and these functions were tested at the required time. The test results are shown in Figure 9. It can be seen from the figure that the time required to query contracts was 0.2~0.3 s, while the time required to register items and create transactions was generally higher than 0.3 s. This is because when the contract executed the query function, it only involved reading the data on the chain, without modifying the status or executing complex business logic. In contrast, contracts for purposes such as creating transactions and executing transactions may need to write new data according to the logic relationship to judge the process of the contract's execution, which will lead to an increase in execution time. The overall test of the architecture conforms to the actual production environment and can be applied.

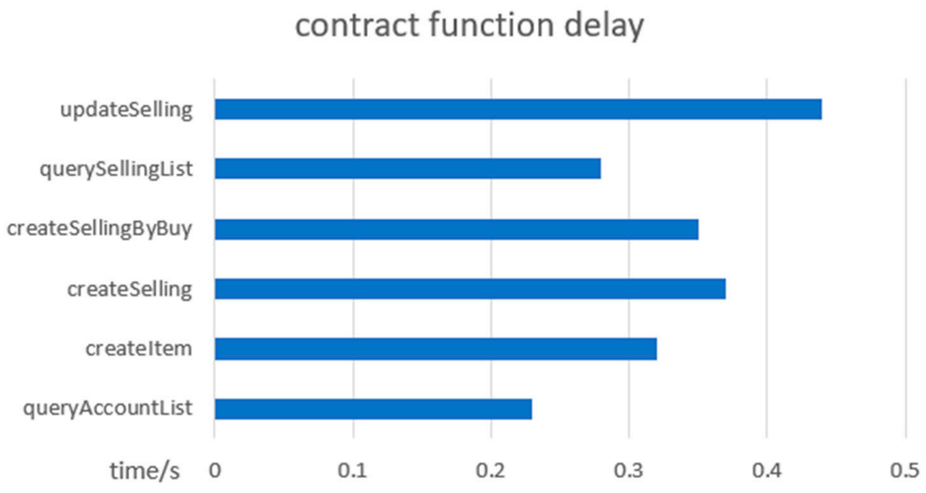


Figure 9. Test results.

4. Summary and Outlook

This paper formulates a smart contract framework for the blockchain transaction field. The contract includes four parts: transaction rules, power and responsibility analysis, a reward and punishment mechanism, and data traceability. This architecture can be applied to transaction-related smart contracts, and allows for corresponding customization and improvement according to the specific application environment. Designing this general architecture can solve the problem of the duplication of labor in the development of current trading smart contracts. The simulation experiment also proves the feasibility of the architecture, which assists in the development of blockchain smart contracts in the transaction field.

In future work, this architecture will be optimized to make it more efficient, secure, and scalable, while considering how to reduce a transaction's execution time, reduce transaction costs, and improve the system throughput. For the future, we propose standardized specifications for smart contracts to promote interoperability and portability between different platforms and applications, as well as the combining of this architecture with actual application scenarios to achieve more complex functions, such as supply chain management, the Internet of Things, digital asset transactions, and other fields, so that smart contracts can be more widely used.

Author Contributions: Conceptualization, Z.L. and W.F.; methodology, C.Z.; software, Z.L.; validation, C.Z., W.F. and Z.L.; formal analysis, C.Z.; investigation, Z.L.; resources, Y.Z.; data curation, Z.L.; writing—original draft preparation, Z.L.; writing—review and editing, W.F.; visualization, C.Z.; supervision, Y.Z.; project administration, Y.Z.; funding acquisition, W.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Major Science and Technology Project of Haikou, under Grant 2020-009, and the Key R&D Project of Hainan province (Grant #: ZDYF2021SHFZ243).

Data Availability Statement: The data used to support the findings of this study are included within the article.

Acknowledgments: The authors would like to thank the editors and the reviewers for their valuable time and constructive comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*. [CrossRef]
2. Shao, Q.; Jin, C.; Zhang, Z.; Qian, W.; Zhou, A. Blockchain Technology: Architecture and Progress. *J. Comput. Sci.* **2018**, *41*, 969–988.
3. Zhu, L. New Trend in Blockchain: From Ethereum Ecosystem to Enterprise Applications. In Proceedings of the 2017 Fourth Global Knowledge Economy Conference, Qingdao, China, 19–21 September 2017.
4. Shi, W. Cloud Service of Blockchain Application System Based on Hyperledger Fabric. Master's Thesis, Zhejiang University, Hangzhou, China, 2018.
5. Luo, X. Research and Implementation of Smart Contract-Oriented Security Development and Debugging Platform. Master's Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2020. [CrossRef]
6. Cai, W.; Luo, Y.; Jiang, S.; Yu, C.; Wu, S. Method and Device for Processing Rule Changes in Smart Contracts. CN Patent CN110400217A[P], 25 April 2019.
7. Wang, P.; Yang, H.; Meng, J.; Chen, J.; Du, X. Formal definition and reference implementation of contract-oriented smart contracts. *J. Softw.* **2019**, *30*, 12.
8. Hu, K.; Bai, X.; Gao, L.; Dong, A. Formal Verification Method for Smart Contracts. *Inf. Secur. Res.* **2016**, *2*, 1080–1089. [CrossRef]
9. Mavridou, A.; Laszka, A. Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach. In *Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2017.
10. Bai, X.; Cheng, Z.; Duan, Z.; Hu, K. *Formal Modeling and Verification of Smart Contracts*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 322–326. [CrossRef]
11. Zheng, X.; Ji, L. Research on Architecture of Digital Bond Trading Contract System Based on Blockchain Technology. *Mob. Inf. Syst.* **2022**, *2022*, 4606902. [CrossRef]
12. Han, D.; Zhang, C.; Ping, J.; Yan, Z. Smart contract architecture for decentralized energy trading and management based on blockchains. *Energy* **2020**, *199*, 117417. [CrossRef]

13. Shi, Y.; Lu, Z.; Tao, R.; Liu, Y.; Zhang, Z. A Trading Model Based on Legal Contracts Using Smart Contract Templates. In *Blockchain and Trustworthy Systems*; BlockSys 2019, Communications in Computer and Information Science; Zheng, Z., Dai, H.N., Tang, M., Chen, X., Eds.; Springer: Singapore, 2020; Volume 1156. [CrossRef]
14. Nazari, M.; Khorsandi, S.; Babaki, J. Security and Privacy Smart Contract Architecture for Energy Trading based on Block chains. In Proceedings of the 2021 29th Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 18–20 May 2021; pp. 596–600. [CrossRef]