International Core Journal of Engineering Volume 5 Issue 11, 2019
ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

Smart Contract Design of Asset Transaction on Blockchain

Xuezhen Zhu [a], Yong Liu, Caili Zhao

Henan University of Science and Technology, Luoyang 471000, China;

[a]2929229533@qq.com

Abstract

Aiming at the problem that traditional asset trading platforms rely on centralized management agencies to complete the transaction process, which can not guarantee the security in the process of asset trading, the characteristics of decentralization, non-tampering modification, observability, autonomy and verifiability of blockchain smart contract technology are utilized. On the basis of blockchain and smart contract theory, this paper establishes the blockchain of asset trading, and develops the asset trading contract system based on the hyperledger experimental environment. The traditional text of asset trading contract is coded, and the storage and execution of smart contract are carried out on the blockchain platform. This not only ensures the asset trading. The security and autonomy of the project also improve the credibility and credibility of the project. The experimental results show that the system can effectively carry out asset trading activities.

Keywords

Blockchain, smart contract, asset transaction, hyperledger, contract verification.

1. Introduction

With the progress of society and the development of economy, asset trading activities become more and more frequent. Traditionally, due to the distrust between users, a trusted third-party platform is needed to support the transaction between users. Users upload their data to the third-party transaction center, which stores all the information in the transaction. Users have third-party platforms to transfer and query assets. In this way, it is easy to appear unsafe problems such as tampering with transaction information.

There are two main reasons for the unsafe problems in traditional asset trading: 1) For human reasons, traditional asset trading activities seem to be handed over to a trusted third-party platform, but managers can not be ignored for negligence or for some benefit to drive unfavorable user data security behavior. User data security mainly depends on the integrity of third-party platforms; 2) For technical reasons, traditional transaction information storage systems mostly use relational databases to store data. Although such database storage systems can solve the problem of data loss due to computer failures, such as network failures, data recovery can be used. Complex technology restores complete data, but once such database system is attacked by malicious external attacks and tampers with transaction data, traditional database system is difficult to deal with; in traditional asset trading activities, data is only protected by simple storage and encryption, and there is no reliable security mechanism in the transaction process, such as signature. Verification mechanism to ensure data security in the transmission process[1].

Blockchain and smart contract technology can solve the problems that traditional technology can not solve well. Firstly, blockchain uses cryptographic method to generate a series of data blocks with correlation between front and back. Each block contains the hash value of the previous block and the

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

hash value of the transaction information of the block, each of which has its own hash value[2]. Blocks of blocks contain transaction information for a period of time. Blockchain is essentially a decentralized distributed database. A little change of block data will lead to the change of blockchain state. In blockchain system, the change of block data can be found quickly and the data can be corrected quickly. This shows that the data stored in blockchain can not be tampered with, which is very important. To a certain extent, it ensures the security and authenticity of the stored data. For activities requiring high credibility such as asset trading, the use of blockchain technology can improve the credibility of the trading system. Secondly, the smart contract on blockchain realizes the business logic in the process of asset transaction through the code on the chain. The business flow of asset transaction is transformed into the code of smart contract. The code and state are stored in the blockchain and executed by the blockchain. Thus, the code itself not only has unchangeable credibility, but also makes the asset possible. Trading business has the characteristics of observability and verifiability, which improves the credibility of trading system[3]. Thirdly, as a distributed storage architecture, blockchains can set different privileged nodes for different users, participate in or partially participate in management, publish authoritative authentication or messages, and these contents are traceable and unchangeable[4]. Finally, blockchains can form chain-in-chain and interconnection chains according to unified standards, realize communication between chains and chains, and solve the problem of mutual communication between different platforms.

Blockchain compiles the business logic of asset transaction into smart contract code, stores the code and state on blockchain, and executes the code on blockchain. The transfer and query of assets can be controlled directly by smart contract, which not only guarantees the security and mandatory of transaction system, but also ensures the execution process of transaction. With the evidence, the credibility of the trading system has been greatly improved.

Firstly, this paper introduces the underlying technology foundation of smart contract-blockchain technology; secondly, it outlines the definition of smart contract, the whole life cycle and the state transition model; thirdly, it builds the hyperledger experimental environment, establishes the asset trading blockchain and develops an smart contract trading system; lastly, it describes the functions of the system. Experiments were carried out to verify the results.

2. Related Work

2.1 Blockchain Technology

Blockchain technology can be traced back to 2008. As the underlying technology of Bitcoin, it was first proposed by Satoshi Nakamoto in the white paper Bitcoin: A Point-to-Point E-Cash System. blockchain in the white paper is described as a way of linking data blocks in chronological order. Synthesize specific data structures and share the general ledger in a non-tampering and non-forgery centralized manner, which is guaranteed by cryptography[5]. As the first blockchain application, Bitcoin solves the double-flower problem and Byzantine general problem in the field of traditional digital encryption currency[6].Blockchain technology does not require a third-party trusted organization. It uses digital encryption, distributed data storage, point-to-point transmission and consensus mechanism to achieve a decentralized trusted system. In a narrow sense, the blockchain is a distributed account in a decentralized system[7], which records all the information since the system was created. Generally speaking, blockchain is a new distributed architecture and computing paradigm that uses digital encryption technology for data transmission and secure access, uses smart contract technology to program and operate data, uses consensus algorithm among nodes to update data, and uses blockchain data structure to store and verify data. Centralization, trustworthiness, non-tampering, programmability, transparency of rules and so on[8]. Blockchain structure is shown in Figure 1. Each block is divided into two parts: block head and block, which involve technical elements such as time stamp, Merck tree, hash algorithm and chain data structure[9].

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

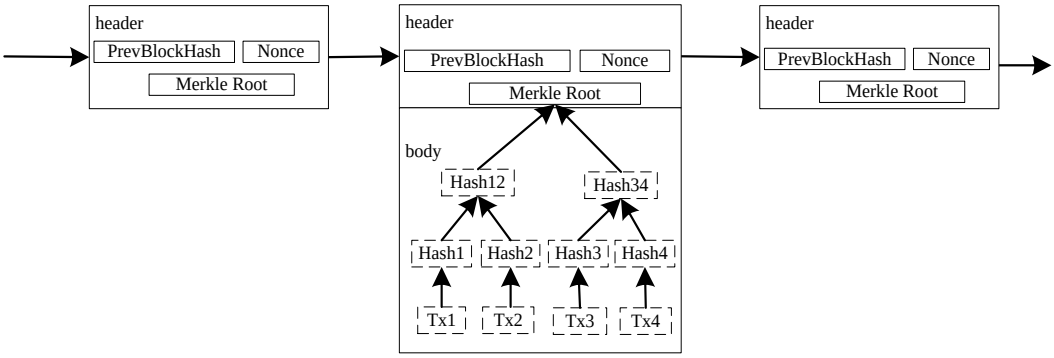

Figure 1. Blockchain structure

Blockchains can be divided into three stages of development. Blockchain 1.0 supports virtual digital currency applications, mostly for payment. Its characteristics include open source code, asymmetric encryption, book sharing and block-chain data structure. The representative application has Bitcoin, but the programmability of such currency is limited, it can not support more complex transactions, and on this basis, it can not create more complex distributed applications. Blockchain 2.0 supports applications other than currencies[10], mostly involving the financial sector. Its characteristics include smart contract, virtual machines and distributed applications. The representative application is ETF, whose essence is to use blockchain as infrastructure to support smart contract applications. Nowadays, blockchain applications are mainly in the 1.0 and 2.0 stages. Blockchain 3.0 supports decentralized self-organization and application beyond the scope of money and finance[11], and is committed to building a distributed society.

Blockchains can be divided into public chains and license chains according to different design systems and application scenarios, and license chains can be divided into alliance chains and private chains. There is no user authorization mechanism in the public chain. Each node can freely join and exit the blockchain network, and participate in the reading and writing of data. The network is interconnected by using a flat topology. There are no central nodes in the network. Each node in the alliance chain has its corresponding entity, so authorized nodes can join and exit the blockchain network, view relevant information according to their authority, form interest-related alliances among organizations in the alliance chain, and maintain the healthy operation of the blockchain together[8]. The write permission of each node in the private chain is controlled internally, and the read permission is open selectively according to the demand. The private chain still has the general structure of blockchain, which is suitable for the audit and management of internal data in a specific organization.

The characteristics of the blockchain, such as decentralization, transparency of rules, non-tampering, distrust and collective maintenance, provide a safe and reliable environment for the recording and execution of smart contract[12]. Firstly, blockchain can establish a consistent representation of all past and current digital events, such as assets and behaviors, without the involvement of third-party trusted institutions and without sacrificing user privacy. Secondly, blockchain provides Turing complete scripting language for smart contract writing, which enhances the flexibility of smart contract application[13], and makes more advanced smart contract application based on blockchain possible.

2.2 Smart contract

Smart contract can be traced back to 1995. Nick Szabo first proposed the concept of smart contract: smart contract is a computational transaction protocol that implements the terms of the contract[15]. At the same time, he also gives the characteristics of smart contract: consistency, enforcement, observability and verifiability. In 1997, Nick Szabo defined smart contract as a set of digital-defined commitments, including agreements on which contract participants can implement their commitments. After the emergence of blockchain, people found that blockchain can provide a credible environment for the execution of smart contract, making the technology of smart contract

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

become the focus of attention in all walks of life. In 2016, the White Paper on the Development of blockchain Technology and Applications in China, published by the Ministry of Industry and Information Technology of China[16], defined the smart contract as an automatically executable program deployed on the blockchain, including programming language, compiler, virtual machine, time, state machine, fault-tolerant mechanism, etc. In a narrow sense, an smart contract is an event-driven, stateful, distributed deployment computer program that program related business logic, law and human protocols. Broadly speaking, smart contract is a computer protocol that can be self-executed and verified after deployment.

Similar to traditional contract, the whole life cycle of smart contract has three stages: contract generation, contract publication and contract execution. As shown in Figure 2, the contract generation stage mainly includes negotiation between the parties involved, clarification of their rights and obligations, formulation of contract standards, programmed of standard contract, and contract verification. The standard contract code is generated. In order to protect the validity and security of the contract, contract participants need to sign the contract. The signed contract is sent to each participating node in the form of P2P. The node will package the contract received over a period of time into a set, calculate the hash value of the set, and transmit its hash value. When broadcasting to other nodes, the node compares the hash values from other nodes with its own hash values. After sending and comparing several times, the nodes reach a consensus on the new contract, and the consensus set of contract is propagated to each node in block form. Contract execution is event-driven. smart contract on blockchain contain state machines, transaction processing mechanisms and storage mechanisms, which are used to receive and process different smart contract. The trigger condition and status of the contract will be traversed periodically. The contract meeting the trigger condition will join the queue of the contract to be verified and be sent to each node. The node will verify the validation contract by signature. The validated contract will be executed after consensus. After execution, the smart contract status will be updated accordingly, and the contract will be integrated. Each process is automatically completed by an smart contract system built in at the bottom of the blockchain[12].

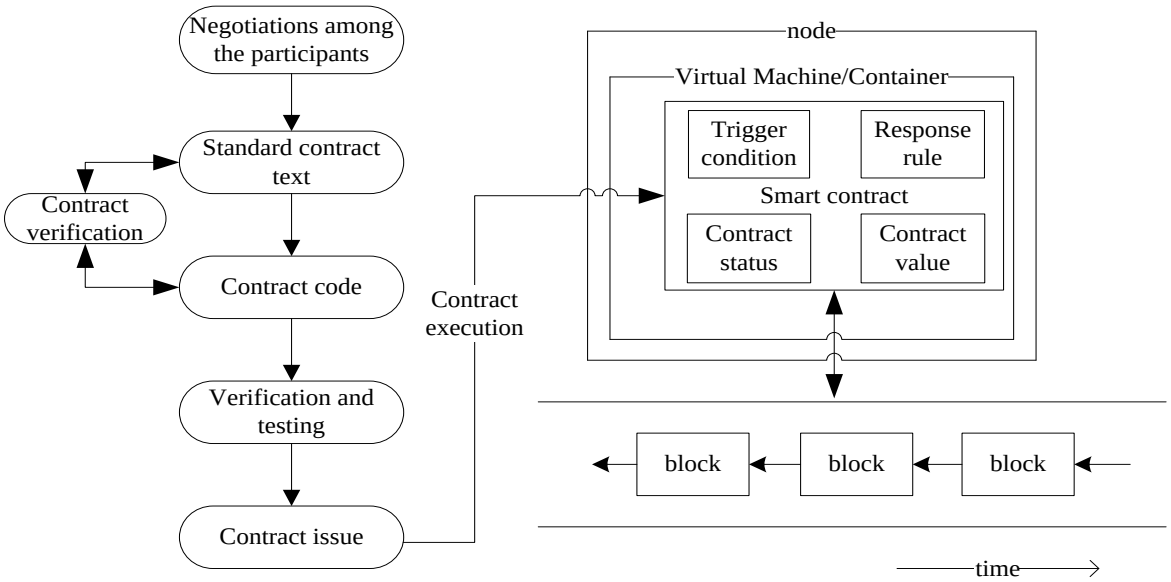

Figure 2. Life cycle diagram of smart contract

Smart contract on blockchain contain a complete state machine, which changes every transaction from the initial state[3]. State refers to any data related to the real world that can be represented by a computer, such as account balance, account reputation value, etc.

The state transition model is as follows:

$$S_{t+1} = f(...f(f(S_t, T_0), T1), ...) \quad (1)$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

Among them, f is the state transition function, S_t is the state at t time, T is the transaction.In a block, there is not only a set of transactions over a period of time, but also a state machine state up to the block. All transactions will be recorded in the blocks of the blockchain to facilitate future historical transaction records query.

Nowadays, although smart contract technology has not been widely used, its technical advantages have been recognized by the majority of relevant researchers. At present, there are ethereum[17], hyperledger[18] and other blockchain platforms supporting smart contract technology. They all have Turing's complete development scripting language, which enables blockchain to support more types of smart contract applications[19]. This paper designs a smart asset trading contract application based on blockchain based on hyperledger platform.

3. Design of Asset Trading System

In this section, based on the theory of blockchain and smart contract, an experimental environment of hyperledger Fabric alliance chain is built to realize a smart asset trading contract. The experimental environment is: CPU is Inter (R) Core (TM) i5-2310, main frequency is 2.90 GHz, memory is 8G, operating system is Linux, programming language is mainly go language.

3.1 Alliance Chain Design

Based on the knowledge of blockchain and smart contract, this paper builds an experimental platform of hyperledger alliance chain, and realizes a smart asset transaction contract based on blockchain. The establishment of alliance chain mainly includes two parts: configuration of start-up node and creation block. The configuration of the start-up node is realized by writing the start-up file. The key parameters of the start-up file in the system are shown in Table 1. The setting of Genesis Block is essentially the setting of block parameters. According to the Genesis Block Framework of hyperledger, the main parameters are set as shown in Table 2.

Table 1. Key parameters of start-up file

parameter	describe
initialization	Initialization, creation of Genesis Blocks
port	Network listening interface
console	Start command line, execute code
RPC	Start remote procedure call communication to deploy and debug smart contract
network id	Set the current network id to distinguish different networks
data directory	Setting the storage path of the current block data
identity	Blockchain identifier used to indicate the current network name
RPCAPI	Setting remote interface between client and database

Table 2. Main parameters of block

parameter	describe
height	Height of block
previous hash	Hash value of front block
root hash	Transaction data root hash value

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

state root	Status roots of account status data
receipts root	Receipt root of transaction execution log
difficulty	Difficulty in block generation
time stamp	Setting block time stamps
nonce	Random numbers for block generation
extra data	Additional information for blocks
number	Number of transactions in the current block
version	Version number

3.2 Smart contract Development

The organizational structure of the asset trading smart contract system is shown in Figure 3. The system mainly includes five modules: user management module, chain code module, security module, exception event processing module and blockchain module.

User management module	Register login	Identity authentication	Key management	Account management
	Smart contract API			
Chain code module	Chain code generation		Chain code execution	
Security module	Asset management		Trusted computing	
Exception event processing module	Abnormal stop		Rollback to normal	
	Blockchain API			
Blockchain module	Communication protocol	Consistency module	Blockchain storage	Blockchain management

Figure 3. Asset trading blockchain architecture

User management module: Users include ordinary users and nodes in the blockchain. Different users have different rights. The module mainly includes user registration, user identity authentication, user key and account management.

Chain code module: manage the whole life cycle of contract. The module mainly includes chain code generation, chain code publishing and execution. Chain code execution means that when a blockchain receives trigger conditions for chain code execution, it executes chain code based on contract status, external information and transaction content, and queries information through the smart contract interface provided by the blockchain and returns the execution results to the blockchain. The main interface of the smart contract provided by the blockchain is shown in Table3.

Table 3. Smart contract interface provided by blockchain

Interface name	input	output
getTraderID	Null	Trader ID
verifySignature	cert,signature,message	Verification results
getState	Key	State value
getTradeID	Null	Transaction number ID

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

putState	Key	Null
recoverState	Key	Null

Security module: for asset management, to ensure the validity, security and credibility of contract computing. The module mainly includes asset management and trusted computing.

Exception event processing module: It is used to terminate the execution of the contract under the exception event and restore the normal state after the exception event. The module mainly includes exception handling and normalization. The core arithmetic flow of asset trading smart contract is shown in Table 4. Firstly, the business rules of the project are converted into contract codes and distributed on the blockchain platform. After the issuance, asset transactions begin, and all parties begin to transfer or inquire about assets. If the assets trading activities conform to the preset rules, the assets trading will succeed, adding the successful transaction records to the blocks, changing the balance of assets of all parties in the assets trading, and changing the status of the blockchain; otherwise, if the assets trading fails, the successful transaction records will not be added to the blocks, and the balance of assets of all parties in the assets trading will remain unchanged.

Table 4. Smart contract algorithms for asset transactions

Algorithms: asset trading process

Input: Trader ID, Trading Volume

Output: If the transaction succeeds, send the transaction request to the smart contract; if the transaction fails, end the transaction.

```
Deployment of Smart Contract
Access Smart Contract and Start Trading
Input trader ID and transaction volume
Get the status value of the participant in the transaction
IF did not find trader ID or status value THEN
  break;
ELSE IF Transaction volume format is numeric THEN
    IF Transaction Sender Balance > Transaction Volume THEN
      Transaction sender balance = Transaction sender balance - Transaction amount;
      Transaction receiver balance = transaction receiver balance + transaction amount;
    END IF
    IF Successful trade THEN
      Increase the successful transaction record in the block and keep the latest status.
    ELSE
    Without adding successful transaction records to the block, the account balance remains unchanged.
    END IF
  END IF
END IF
```

Blockchain module: used to store all transaction information and manage blockchain. The module mainly includes communication protocol, consistency module, blockchain storage and blockchain management.

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

4. Exprerimental test

Hyperledger is an open source, block-chain data structure-based, smart contract-enabled underlying system. In the hyperledger system, smart contract are equivalent to programs deployed on blockchain. Use containers in hyperledger to execute smart contract written. Based on the experimental environment of hyperledger blockchain, this paper implements a smart asset trading contract.

This section is mainly divided into the following parts: contract compilation, contract deployment, contract validation and exception handling.

Contract compilation. The contract defines the business logic of asset transaction and the business rules of querying status data. This paper uses go language to write the smart contract of asset transaction. The implementation of the contract algorithm is shown in Figure 4. In order to prevent malicious contract from directly threatening the security of blockchain nodes, contract can not run directly on blockchain nodes, and need to run in isolated sandbox environment. In this paper, docker container is used as the operating environment of contract.

```
Avalbytes, err := stub.GetState(A)
if err != nil {return shim.Error("Failed to get status value!")}
if Avalbytes == nil {return shim.Error("No corresponding entity was found!")}
Aval, _ = strconv.Atoi(string(Avalbytes))
Bvalbytes, err := stub.GetState(B)
if err != nil {return shim.Error("Failed to get status value!")}
if Bvalbytes == nil {return shim.Error("No corresponding entity was found!")}
Bval, _ = strconv.Atoi(string(Bvalbytes))
X, err = strconv.Atoi(args[2])
if err != nil {return shim.Error("Invalid transaction,transaction volume format must be numric")}
if X >= Aval {return shim.Error("Invalid transaction,the amount of transaction should not be
greater than the available balance of the account")}
Aval = Aval - X
Bval = Bval + X
fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)
err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
if err != nil {return shim.Error(err.Error())}
err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
if err != nil {return shim.Error(err.Error())}
return shim.Success(nil)
```

Figure 4. The implementation of the contract algorithm

Contract deployment. Compiled to verify the correctness of the smart contract, and deploy it on the blockchain. This experiment creates two hyperledger accounts, account A and account B, in which A and B can be both senders and receivers of asset transactions. This paper deploys the compiled asset trading smart contract on the hyperledger blockchain and validates its functions.

Contract validation. Contract validation is mainly functional validation, mainly including the following aspects: 1) Query the status data of asset trading participants, such as account balance. 2) The participants in asset transactions conduct asset transactions. 3) Verify that the transaction results are correct after the asset transaction is completed. In this case, the contract validation interface is shown in Figure 5. Firstly, the user's account balance is queried by contract: the local authentication mechanism (MSP) is obtained, the signature identification is obtained, and the chain code command parameters are checked by ESCC (endorsement system chaincode) and VSCC (verification system

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

chaincode). The balance of accounts of users a and B is 100 and 200 assets respectively. Then, the smart contract user B is invoked to send 30 assets to a. After the asset transaction proposal is verified, endorsed and agreed, the asset transaction is completed, the valid asset transaction is written into the blockchain, and the status database is updated. After that, the balance of accounts of users a and B is 130 and 170 assets respectively through contract query, and the asset transaction is successful. The verification results show that the above functions have been verified, and the system is feasible.

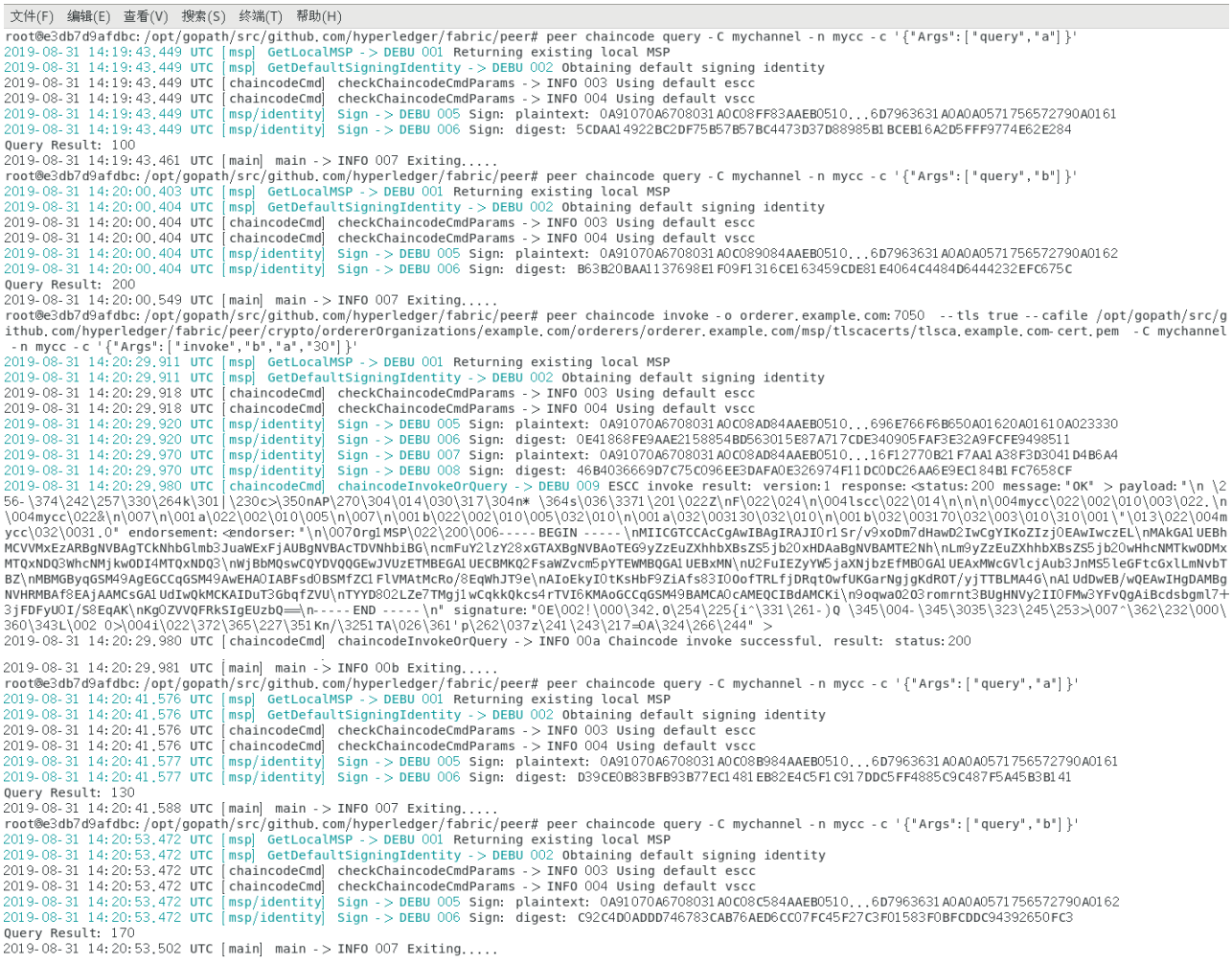

Figure 5. Functional verification of smart contract for asset transactions

Abnormal event handling. Abnormal event handling mainly includes controlling and prompting the illegal or wrong input of the user in the contract to prevent the system from collapse. It mainly includes the following aspects: 1) querying the status data of asset trading participants, such as account balance. 2) The participants in asset transactions conduct illegal or wrongful asset transactions. 3) Verify the success of the transaction after the completion of the asset transaction. In this case, the operation interface of contract exception event processing is shown in Figure 6. Firstly, the account balance of user a and B is 130 and 170 assets respectively by contract query. Then, the smart contract user a is invoked to send t01 assets to b, the node endorsement fails, and the system prompts "invalid transaction, the transaction amount format must be numerical type". Call smart contract user a to send 131 assets to b, node endorsement failed, the system prompted "invalid transactions, transactions can not be greater than the account balance". Finally, through the contract query user a and B account balances are 130 and 170 assets, account balances remain unchanged, illegal or erroneous asset transactions failed to take effect. The verification results show that the above abnormal events can be effectively controlled and prompted, and the system is feasible.

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

```
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)
Query Result: 170
2019-08-31 14:20:53.502 UTC [main] main -> INFO 007 Exiting.....
root@e3db7d9afdbc:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050  --tls true --cafile /opt/gopath/src/g
ithub.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem  -C mychannel
 -n mycc -c '{"Args":["invoke","a","b","t01"]}'
2019-08-31 14:21:22.511 UTC [msp] GetLocalMSP -> DEBU 001 Returning existing local MSP
2019-08-31 14:21:22.511 UTC [msp] GetDefaultSigningIdentity -> DEBU 002 Obtaining default signing identity
2019-08-31 14:21:22.522 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default escc
2019-08-31 14:21:22.522 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default vscc
2019-08-31 14:21:22.523 UTC [msp/identity] Sign -> DEBU 005 Sign: plaintext: 0A91070A6708031A0C08E284AAEB0510...6E766F6B650A01610A01620A03743031
2019-08-31 14:21:22.532 UTC [msp/identity] Sign -> DEBU 006 Sign: digest: 9DE089AA8C83B73C8BE2F3EDD154A1E994BB1C601CD618A14480B0AED577377C
Error: Error endorsing invoke: rpc error: code = Unknown desc = chaincode error (status: 500, message: Invalid transaction, transaction volume format must be
numeric) - <nil>
Usage:
  peer chaincode invoke [flags]

Flags:
  -C, --channelID string   The channel on which this command should be executed (default "testchainid")
  -c, --ctor string        Constructor message for the chaincode in JSON format (default "{}")
  -n, --name string        Name of the chaincode

Global Flags:
      --cafile string              Path to file containing PEM-encoded trusted certificate(s) for the ordering endpoint
      --logging-level string       Default logging level and overrides, see core.yaml for full syntax
  -o, --orderer string             Ordering service endpoint
      --test.coverprofile string   Done (default "coverage.cov")
      --tls                        Use TLS when communicating with the orderer endpoint
  -v, --version                    Display current version of fabric peer server

root@e3db7d9afdbc:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050  --tls true --cafile /opt/gopath/src/g
ithub.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem  -C mychannel
 -n mycc -c '{"Args":["invoke","a","b","131"]}'
2019-08-31 14:22:12.165 UTC [msp] GetLocalMSP -> DEBU 001 Returning existing local MSP
2019-08-31 14:22:12.165 UTC [msp] GetDefaultSigningIdentity -> DEBU 002 Obtaining default signing identity
2019-08-31 14:22:12.171 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default escc
2019-08-31 14:22:12.171 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default vscc
2019-08-31 14:22:12.172 UTC [msp/identity] Sign -> DEBU 005 Sign: plaintext: 0A90070A6608031A0B089485AAEB0510...6E766F6B650A01610A01620A03313331
2019-08-31 14:22:12.172 UTC [msp/identity] Sign -> DEBU 006 Sign: digest: 1EEC33624DAC7ABCAE6509C7F33FC992DDAE931D748583E65DEFDCB0793674C4
Error: Error endorsing invoke: rpc error: code = Unknown desc = chaincode error (status: 500, message: Invalid transactions, the amount of transactions should
 not be greater than the available balance of the account) - <nil>
Usage:
  peer chaincode invoke [flags]

Flags:
  -C, --channelID string   The channel on which this command should be executed (default "testchainid")
  -c, --ctor string        Constructor message for the chaincode in JSON format (default "{}")
  -n, --name string        Name of the chaincode

Global Flags:
      --cafile string              Path to file containing PEM-encoded trusted certificate(s) for the ordering endpoint
      --logging-level string       Default logging level and overrides, see core.yaml for full syntax
  -o, --orderer string             Ordering service endpoint
      --test.coverprofile string   Done (default "coverage.cov")
      --tls                        Use TLS when communicating with the orderer endpoint
  -v, --version                    Display current version of fabric peer server

root@e3db7d9afdbc:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
2019-08-31 14:22:35.220 UTC [msp] GetLocalMSP -> DEBU 001 Returning existing local MSP
2019-08-31 14:22:35.220 UTC [msp] GetDefaultSigningIdentity -> DEBU 002 Obtaining default signing identity
2019-08-31 14:22:35.220 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default escc
2019-08-31 14:22:35.220 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default vscc
2019-08-31 14:22:35.230 UTC [msp/identity] Sign -> DEBU 005 Sign: plaintext: 0A90070A6608031A0B08AB85AAEB0510...6D7963631A0A0A0571756572790A0161
2019-08-31 14:22:35.230 UTC [msp/identity] Sign -> DEBU 006 Sign: digest: AB1B2334895C0FB579EAC65869A76B9116B324AEBC28353871C8D783A2D029E3
Query Result: 130
2019-08-31 14:22:35.255 UTC [main] main -> INFO 007 Exiting.....
root@e3db7d9afdbc:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n mycc -c '{"Args":["query","b"]}'
2019-08-31 14:22:43.605 UTC [msp] GetLocalMSP -> DEBU 001 Returning existing local MSP
2019-08-31 14:22:43.605 UTC [msp] GetDefaultSigningIdentity -> DEBU 002 Obtaining default signing identity
2019-08-31 14:22:43.605 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default escc
2019-08-31 14:22:43.605 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default vscc
2019-08-31 14:22:43.606 UTC [msp/identity] Sign -> DEBU 005 Sign: plaintext: 0A91070A6708031A0C08B385AAEB0510...6D7963631A0A0A0571756572790A0162
2019-08-31 14:22:43.606 UTC [msp/identity] Sign -> DEBU 006 Sign: digest: 5475CEECCA709AD5EC48259A56579491E30804B403EAA906DE28CBE08F0FCAB0
Query Result: 170
```

Figure 6. Abnormal processing of smart contract in asset transaction

From the above experimental results, we can see that the smart contract of asset trading can basically achieve the expected design goals, and the experiment also verifies the feasibility and effectiveness of the smart contract of asset trading. The smart contract of asset transaction proposed in this paper takes blockchain as the bottom environment, develops smart contract on blockchain, and manages asset transaction system spontaneously according to preset protocol, and does not allow unilateral modification or termination of protocol, which effectively reduces the risk of human intervention. In this paper, the advantage of asset trading smart contract is to build an asset trading smart contract platform with blockchain as the carrier. blockchain technology is used to construct a distributed consistency standard in network. Using the characteristics of blockchain, such as non-tampering, observability, traceability and verifiability, we can improve the information transparency of asset trading process and the credibility of asset trading system.

5. Conclusion

Based on the experimental environment of hyperledger blockchain and the theory of smart contract, this paper develops an asset trading system. The feasibility and validity of the system are verified by experiments. The next step is to continue to improve the function of the asset trading system and

ISSN: 2414-1895 DOI: 10.6919/ICJE.201910_5(11).0012

enhance the practicability and robustness of the system. Nowadays, the development of blockchain has gradually changed from digital currency stage to smart contract stage. In the next few years, smart contract technology will still be a research and application popular in the field of blockchain. However, smart contract technology based on blockchain is still in its early stage of development and faces many challenges. How to coordinate security, efficiency and decentralization? The relationship between the three needs further study. Although these problems exist, many researchers are trying to improve them. smart contract technology based on blockchain has broad application prospects because of its unique advantages.

References

[1] Han Shuang, Pu Baoming, Li Shunxi, etc. Application of Block Chain Technology in Digital Asset Security Transaction [J]. Computer System Application, 2018, 27 (03): 205-209.

[2] Calvaresi D , Dubovitskaya A , Calbimonte J P , et al.Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review[C]// International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer, Cham, 2018.

[3] Huang Jiehua, Gao Lingchao, Xu Yuzhuang, et al. Intelligent contract design on crowdsourcing block chain [J]. Information security research, 2017,3(03): 211-219.

[4] Chen Zhidong, Dong Aiqiang, Sun He, etc. Private Block Chain Research Based on Crowdsourcing [J]. Information Security Research, 2017, 3 (03): 227-236.

[5] Liu Y , Li R , Liu X , et al. An efficient method to enhance Bitcoin wallet security[C]// 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID). IEEE, 2017.

[6] Zhou Guoliang, Lu Linjie. Application of Block Chain Technology in Energy Internet [C]//2016 Annual Conference on Power Industry Informatization, Tianjin, China, 2016.

[7] Li C , Palanisamy B . Decentralized Release of Self-Emerging Data using Smart Contracts[C]// 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS). IEEE, 2018.

[8] Ma Chunguang, An Jing, Bi Wei, et al. Intelligent contracts in block chains [J]. Information network security, 2018 (11): 8-17.

[9] Jiang Haifeng. Research on the Application Model of Block Chain Technology in Financial Industry [D]. Zhejiang Hangzhou, Zhejiang University, 2018.

[10] Ulieru M. Blockchain 2.0 and Beyond: Adhocracies[M]//Banking Beyond Banks and Money. 2016.

[11] Peters D , Wetzlich J , Thiel F , et al. Blockchain applications for legal metrology[C]// IEEE International Instrumentation & Measurement Technology Conference. IEEE, 2018.

[12] He Haiwu, Yan'an, Chen Zehua. Overview of Intelligent Contract Technology and Application Based on Block Chain [J]. Computer Research and Development, 2018, 55 (11): 2452-2466.

[13] Parizi R M, Amritraj, Dehghantanha A. Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security [C]// International Conference on Blockchain. Springer, Cham, 2018.

[14] Watanabe H, Fujimura S, Nakadaira A, et al. Blockchain contract: Securing a blockchain applied to smart contracts [C]//IEEE International Conference on Consumer Electronics. 2016.

[15] Wang Chunyu, Zhang Shoukun. Intelligent and Financial Contracts [J]. Shang, 2016 (06): 198.

[16] Zhang Yutian. Some ideas and suggestions on the development of block chain industry [J]. China International Finance and Economics (English and Chinese), 2017 (20): 25-26.

[17] Fairley P. Ethereum will cut back its absurd energy use[J]. IEEE Spectrum, 2018, 56(01):29-32.

[18] Benhamouda F, Halevi S, Halevi T. Supporting Private Data on Hyperledger Fabric with Secure Multiparty Computation[C]// 2018 IEEE International Conference on Cloud Engineering (IC2E). 2018.

[19] Shao Qifeng, Jin Chengqing, Zhang Zhao, et al. Block Chain Technology: Architecture and Progress [J]. Journal of Computer Science, 2018, 41 (05): 969-988.