ATLANTIS PRESS

Advances in Engineering Research, volume 149

2nd International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018)

Research and Application of Warehouse Receipt Transaction Based on Smart Contract on the Blockchain

Yafei Chen[1, a], Zhihong Zhang [b, *] and Beibei Yang[2, c]

[1]School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China;

[2]School of Software and Microelectronics, Northwestern Polytechnical University, Xi, an 710072, China.

[a]cyf_zzu90@163.com, [b]zhzhang@esunny.cc, [c]ybb_xgd@163.com

Keywords: warehouse receipt transaction, blockchain, smart contract, distributed ledger, PBFT consensus algorithm.

Abstract. To solve the credibility and supervision lacking problem in the OTC warehouse receipt trading system, a scheme of using smart contract on the blockchain to implement the OTC warehouse receipt trading system is presented. As a distributed ledger, blockchain has the advantages of decentralization, irreversibility and trustworthiness. The warehouse receipt transaction data is recorded on the blockchain, ensuring its safety and credibility. Smart contract is automatically executed code stored on the blockchain, it can directly control the transaction of digital assets. No participant can control and tamper it before the contract is lapsed. In the process of using the Ethereum smart contract to realize warehouse receipt transactions, consensus is reached by the PBFT consensus algorithm, Since the trusted third-party is not needed, the lack of supervision problem is solved. This paper introduces the application of blockchain and smart contract to the OTC warehouse receipt trading system. It includes the design of smart contracts, the verification of smart contracts security. This paper also verifies the feasibility of designing smart contracts by using the instance of listed transactions.

1. Introduction

Warehouse receipt transaction is a dealer on the OTC warehouse receipt trading platform through the listing or negotiation transactions to sell warehouse receipts to the dealer who needs of the warehouse receipts and complete the business of warehouse receipts transfer and transfer of funds by bilateral commission liquidation or settlement by themselves.

Take the listing transaction as an example to introduce the process of warehouse receipt transaction in the OTC warehouse receipt trading system: the seller dealer issues the warehouse receipt information (including the price, the quantity, the settlement method, etc.) to be sold through the warehouse receipt transaction platform. After the listing is successful, all the warehouse receipts will display on the list of market. At this time, the buyer selects the appropriate warehouse receipt for delisting, and then the Exchange transacts the warehouse receipt and transfer of funds through the settlement method set by the seller.

Analysis of the warehouse receipt transaction process, we can find that there are two main problems in the OTC warehouse transaction system. First, the issue of credibility, how to store the transaction data to ensure the safety and credibility, cannot be tampered with. Second, regulatory issues, the transaction process between users requires third-party audits, which makes the transaction inefficient and consumes a certain amount of calculation cost.

This paper proposes to apply the blockchain and smart contract technology to the OTC warehouse transaction system and use smart contracts to customize the warehouse receipt transaction template. The process of the transaction reached a consensus by the system, without the need for third-party audits and solve the problem of inadequate supervision of OTC systems. The use of blockchain records warehouse receipt transactions, which making the transaction data security and credible, increasing the credibility of the OTC transaction system. This article will mainly introduce the overall

structure of the warehouse receipt transaction system, the design of smart contracts, the interaction process of smart contracts during the warehouse receipt transaction and the security issues of smart contracts.

2. Status of Smart Contract Research

Smart contract is essentially an executable computer program that was proposed by Nick Szabo [1] in 1994. However, the smart contract has not been applied to the actual business system. The first reason is how smart contracts control the physical assets to ensure the effective execution of contracts. Second, smart contracts lack the safe and trustworthy execution environment. The advent of blockchain technology [2] addresses these issues by providing a credible, decentralized execution environment for smart contracts and playing an important role in smart contracts in asset management, contract management, and regulatory enforcement [3].

Smart contracts based on blockchain technology can not only take advantage of cost efficiency, but also avoid the interference of malicious acts in the normal implementation of the contract. The smart contract is written into the blockchain in the form of digitization, which is stored, read and executed by the characteristics of the blockchain technology. The whole process is transparent, traceable and cannot be tampered with. At the same time, a set of state machine system is constructed based on consensus algorithm of blockchain, which makes smart contracts work efficiently [4].

As shown in Table 1, the comparison of smart contract features between Ethereum [5] and Hyperledger [6]:

Table 1 Smart Contracts comparing

	Ethereum	HyperLedger
contract complexity	can not be complicated, based EVM	can complicate, based docker
contract deploy	EVM bytes, send transaction	chaincode, through SDK
contract update	can not be updated	1.0version support update
contract call	send transaction	as transaction
external system interaction	can not external interaction	external system can interact

Other smart contracts include RootStock and Corda.

RootStock is a smart contract distribution platform on the Bitcoin blockchain [7]. Implemented an improved version of Ethereum virtual machine, deploying complex smart contracts as a sidechain. RootStock has the following three main features:

(1) Turing complete and resource-bound virtual machines (smart contracts);

(2) Two-way anchored sidechain (the conversion between root and bitcoin is achieved by bidirectional anchoring);

(3) Dynamic hybrid joint mining (safety consensus) and low latency network (fast payment).

Codar [8] is a distributed platform for recording and processing financial agreements and provides an implementation framework for smart contracts. Have to go to the centralization, support a variety of consensus mechanisms, increase regulatory nodes and other features.

3. The Overall Design of The Warehouse Receipt Transaction System

The warehouse receipt transaction system based on blockchain smart contract is a blockchain system developed on the Ethereum platform. The system hierarchy architecture mainly includes four layers: a physical node layer, a platform layer, a contract layer and an application layer, as shown in Figure 1. Since the PBFT consensus algorithm does not require mining, the node can also maintain the whole network security, so cancel the excitation layer in the original structure of Ethereum.

(1) Physical node layer: the underlying physical node server. According to the particularity of warehouse receipt transactions (minority node accounting) and the characteristics of the PBFT consensus algorithm (a minimum of four nodes can reach a consensus), 4-7 servers are used to form the physical layer.

(2) Platform layer: it, s mainly to improve the structure of Ethereum in order to support the warehouse receipt trading business. Platform layer function of each module are as follows:

Network layer: mainly includes P2P network, message dissemination mechanism, data verification mechanism, to ensure the interaction between nodes.

PBFT consensus layer: Implements the PBFT consensus algorithm, which reach consensus on block data among different nodes.

Data layer: defines the structure of the main data, protocol. For example, the transaction data structure, block data structure, consensus data structure.

Smart Contract Execution Engine: The essence is a virtual machine that executes smart contract code. The execution environment is blockchain data, the result of which is the change in the state of the data on the blockchain.

(3) Contract layer: The smart contract layer mainly uses smart contracts to implement various logic, including business logic and other logic. Business logic can be warehouse receipt transaction logic. Other logic, such as role permissions control, node management and so on.

(4) Application layer: the user interaction with the blockchain system interface. This layer mainly implements two functions: first, the interaction between users and smart contracts; second, the management of user keys.

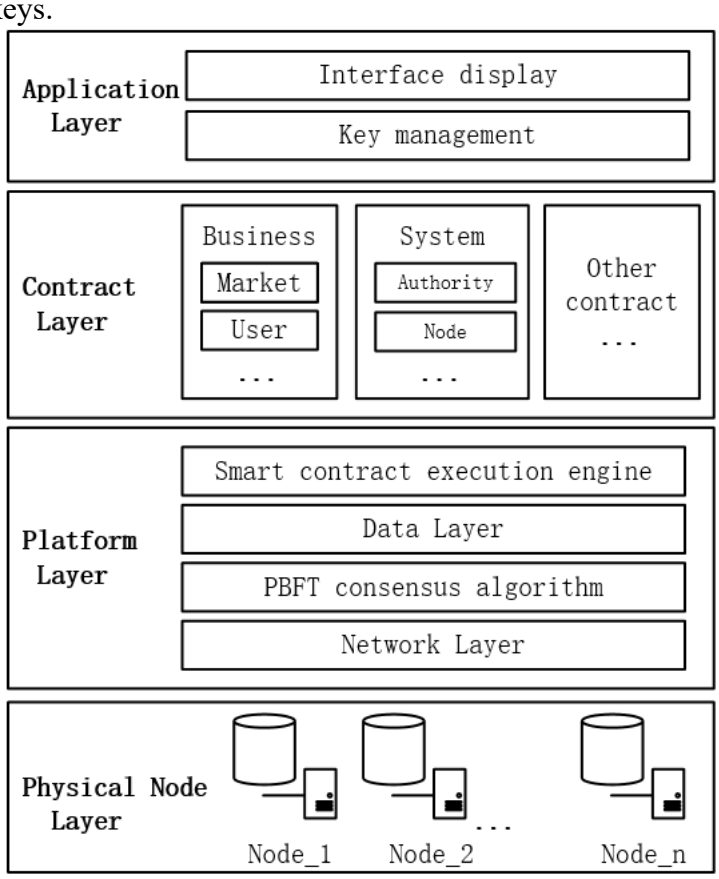

Fig. 1 System architecture diagram

The main features of the overall design of the warehouse receipt transaction system based on the blockchain smart contract are as follows:

(1) Block Cache Design: Ethereum in the original model, the transaction is synchronized to the block transaction needs to be executed, block synchronization to the blockchain, the implementation of the original block transactions again. Warehouse receipts a large number of transactions, will inevitably result in a waste of time. The block caching mechanism is used in combination with the PBFT consensus algorithm, the block generated by the proposal node are cached after the consensus node verifies and reaches a consensus. When the block is inserted into the chain, the consensus block is obtained from the cache first, and if the difficulty value is 0, then re-execute the block transaction for verification, otherwise directly into the chain.

(2) Node Deployment: According to the warehouse transaction business scenario, the node needs carry out permissions management. There are three types of nodes: accounting nodes, which can access non-chargeable nodes and cannot access nodes. Nodes support configuration files and smart contracts for permission management; Nodes do not need to be rebooted when they join and exit; Nodes' change information is recorded on the blockchain.

(3) Key Management: The user's key information is managed by Metamask, stored in the front-end and managed by the user himself. User key information is not on the chain, not open, to ensure the privacy and security of key information.

(4) PBFT Consensus Algorithm [9]: Compared with the existing consensus algorithm of Ethereum POW, the PBFT consensus algorithm has the advantages of higher efficiency, less adaptation node size, high anti-attack ability and application of the private chain or alliance chain scenario. In addition, PBFT algorithm does not have to carry out mining, the node can also maintain the whole network, in line with the design of the program to the currency, a better warehouse receipt trading system.

(5) Smart Contract Design: The smart contract customizes the warehouse receipt transaction record contract code and global status in the block chain. The contract code is executed automatically. The execution process is based on system consensus and the execution result is recorded in the block chain. Not only Improved transaction security without the need for third-party audits.

4. Smart Contract Design

In this paper, the smart contract is mainly designed according to the business process of listing, delisting, forming a transaction in the warehouse receipt transaction, and designing the warehouse receipt transaction as User contract, Market contract, ID Management contract, Library contract and so on. The execution after forming a deal is replaced by the administrator and is designed as a admin contract.

The main functions of each type of contract are as follows:

(1) User contract: the user listed, delisting, the formation of contract transactions and other operations.

(2) Market contract: a summary of all the warehouse receipts information, the seller user through the listing function release warehouse receipt information to trigger the market contract shows the details of the warehouse receipt, the buyer user through the delisting function to buy warehouse receipts, trigger the market contract to modify market information.

(3) ID Management contract: management warehouse receipts listing number, market number and the contract number.

(4) Admin contract: add ordinary users, add warehouse receipts and set funds for ordinary users and so on.

(5) Library contract: such as LibSheetMap contracts, storage warehouse receipts information; LibMarketMap contract, store market information.

The following describes the main contract interactive scene. As shown in Figure 2, the admin contract interaction process. The administrator is the user with the highest authority set by the configuration file, and can perform operations such as adding users, adding warehouse receipts, setting funds and so on.

When an administrator adds a user, the following steps occur:

(1) sending a message that triggers an administrator's external account to its Admin contract;

(2) Admin contract to invoke user contract, generate user instance;

(3) Admin contract checks Userlist contract and register the user.

When an administrator adds warehouse receipts, the following steps occur:

(1) sending a message that triggers an administrator's external account to its Admin contract;

(2) Admin contract to check the Userlist contract, access to user addresses;

(3) send user address information to User contract;

(4) User contract call ID Management contract to obtain the warehouse receipt number;

(5) User contract to obtain the warehouse receipt number, and then call LibSheetMap to stored the warehouse receipt information.

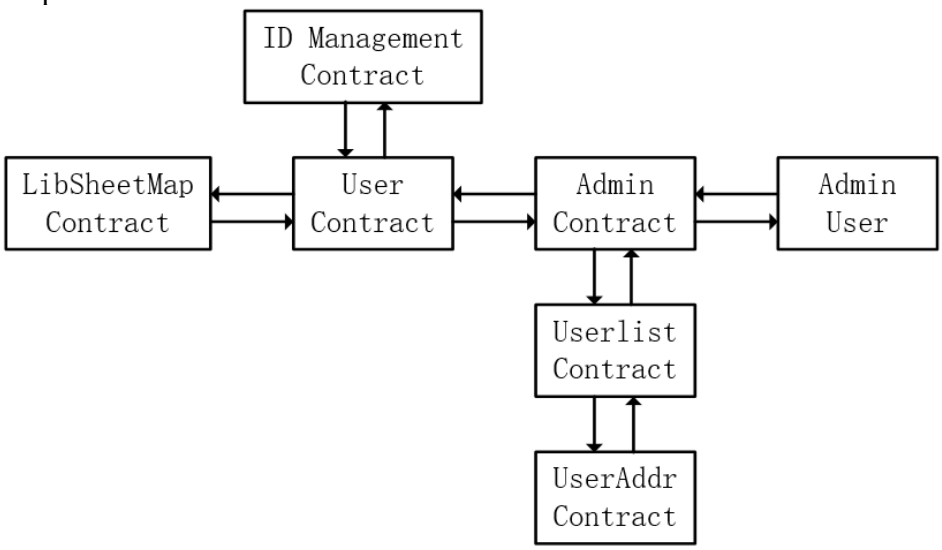

Fig. 2 Admin Contract interaction process

Listing request interactive scene, as shown in Figure 3. Each user who has permissions and is registered by the administrator to the system will have a corresponding User contract. When users sell warehouse receipts, the following steps occur:

(1) choose to sell the warehouse receipt information, send the transaction and trigger the message from the EOA to the User contract;

(2) User contract to send insert warehouse receipt message to Market contract;

(3) Market contract check ID Management contract to obtain market id, then insert warehouse receipt information;

(4) Market contract trigger market update event, return the warehouse receipt serial number to the User Contract;

(5) User contract execution warehouse receipt freeze operation, returns the listing request success message to the user.

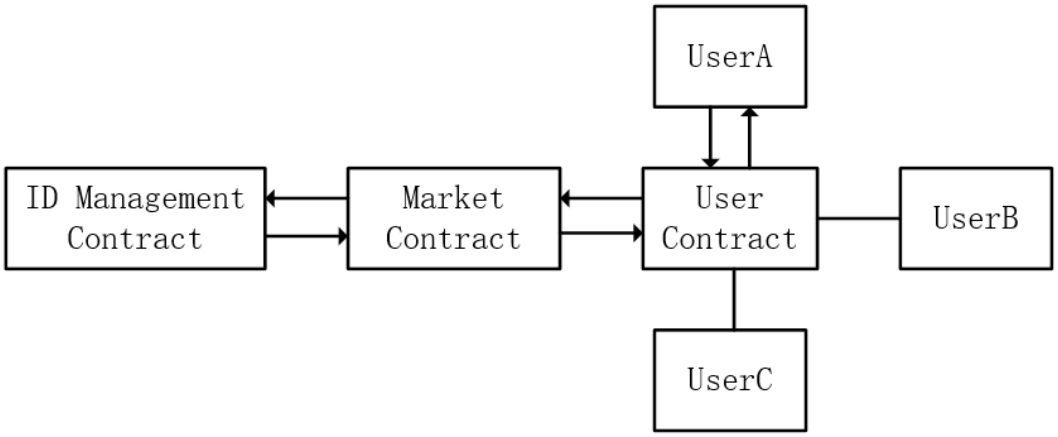

Fig. 3 Listing operation interaction process

Delisting request interactive scene, as shown in Figure 4. Take UserA1 delists UserA2, s warehouse receipt for example. When users delist warehouse receipts, the following steps occur:

(1) UserA1 issues a delisting request transaction to trigger a message from A1 to the user contract;

(2) Check the market contract of A1 for the contract of the user to get the information of warehouse receipt of A2 and execute the delisting;

(3) Market contract to update both warehouse receipts and sales information, call the buyer and seller contract to generate contracts;

(4) A1, s user contract returns delisting request success message to A1 user.

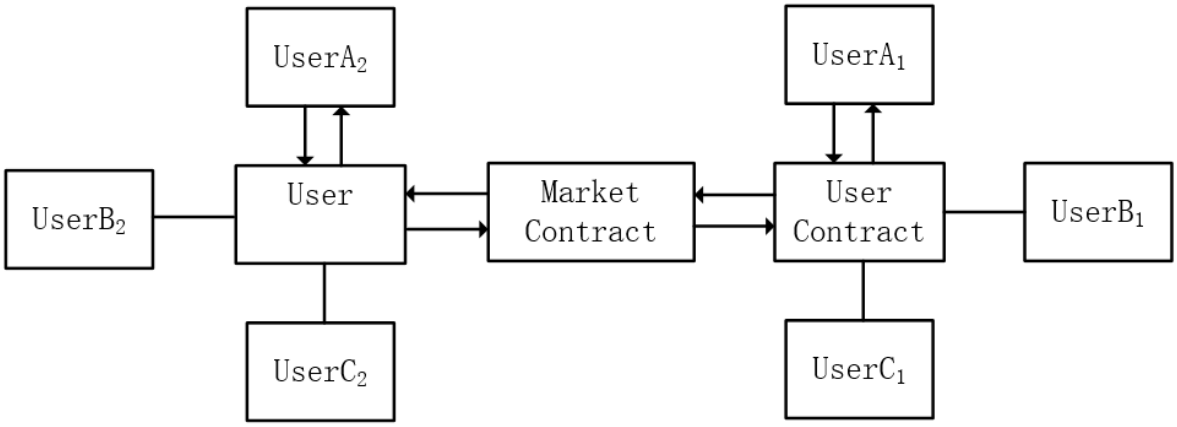

Fig. 4 Delisting operation interaction process

The implementation mechanism of smart contract [10], as shown in Figure 5. The smart contract written by Solidity Language [11] is compiled into binary bytecode by Ethereum client or Remix [12] development tool and then sent to the network through transaction (ie, SendTransaction). After reached consensus by PBFT consensus algorithm, deployed to the blockchain waiting for the message call. There are two ways to interact with smart contracts:

(1) Send Transaction: This method modifies the blockchain data and needs to be agreed on the whole network (same as the deployment contract process). The call is recorded in the blockchain and the result (global state) is stored in the state database (Such as transfer transactions between accounts, the transfer amount recorded in the block chain, the account balance changes stored in the state database).

(2) Message Call: The message call (such as obtaining the block height and querying the balance) only inquires the blockchain data and will not be recorded in the blockchain or not change the internal state of the contract. Smart contracts support event mechanisms for asynchronous calls. By monitoring the contract events, access to event information (such as transaction hash) to verify the transaction execution is successful. Currently, smart contracts can only access intra-link data and cannot actively monitor and respond to off-chain events.

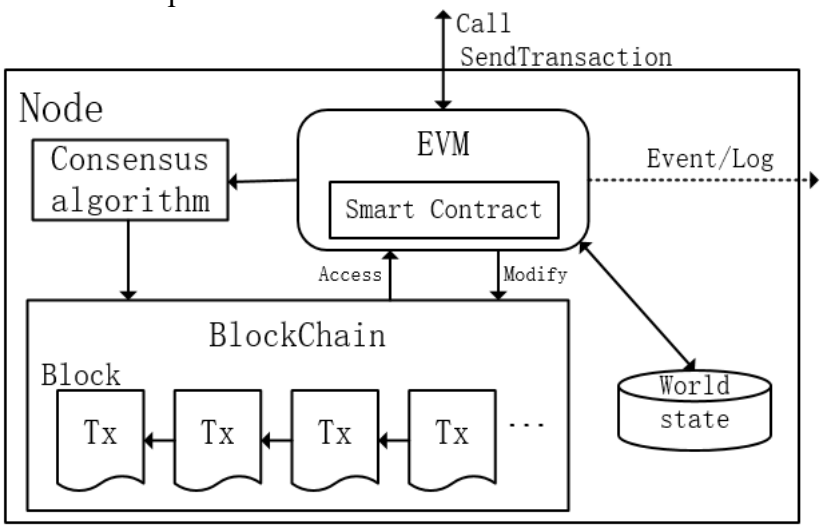

Fig. 5 Smart contract enforcement mechanism

5. Application of Smart Contract

The warehouse receipt transaction system uses the Truffle framework [13] to deploy smart contracts and front-end code. Take the list transaction in warehouse receipt transaction as an example to verify the feasibility of smart contract design. According to the actual warehouse receipt transaction scenario, the users are designed into two types:

(1) Admin user: Admin user has higher authority. The main function is to register users, add warehouse receipts, set funds, and confirm the warehouse receipt transaction request.

(2) Ordinary user: Can execute listing, delisting, check warehouse receipts, view pending orders, view contracts and so on.

The following is the realization of the smart contract interface:

Table 2 User contract

Method name	Features
listRequest	Handling user listing requests
delistRequest	Handling user delisting requests
recordTrade	Create a deal contract
getSheetMap	Get warehouse receipt information
getTrade	Get the contract information

Table 3 Market contract

Method name	Features
intsertMarketMap	Insert market information
getMarketMap	Get market information
updateMarketMap	Update market information

Table 4 Admin contract

Method name	Features
addUser	Add users
insertSheet	Insert warehouse receipt
modifyFunds	Modify funds
confirmList	Confirm the listing request
confirmNeg	Confirm the negotiation request

As mentioned in the previous section, the user executes the operation of modifying blockchain data, such as listing and delisting. By sending the transaction and adding its own signature, the transaction execution results reached a consensus and were recorded on the block chain. The user executes the query operation, such as viewing the warehouse receipt information, viewing the contract information, etc., interacting with the contract through the message call, and the execution result is not recorded on the blockchain.

In this paper, the design and application of smart contracts in warehouse receipt transaction system have been realized and verified. The actual tests show that the smart contract and blockchain to achieve warehouse receipt business, not only exerts the advantage of the smart contracts in efficiency, but also does not require third-party audits and solve the problem of inadequate regulation. The results of the transaction are stored in the blockchain to ensure that the transaction data is security, trustworthy and cannot be tampered with, thus solving the problem of the credibility of the warehouse receipt trading system.

6. Smart Contract Security Verification

Smart contracts are self-executing code stored on the blockchain that does not require third-party audits and no parties can control and transfer contractual assets until the contract lapsed. Smart contracts have certain advantages in terms of efficiency and security, however, the application of smart contracts is still in its infancy, security is still an important problem that cannot be ignored. Currently, there are two serious loopholes in Ethereum that have caused huge losses. Such as TheDAO Contract Vulnerability [14], Parity Multi-Signature Wallet Vulnerability.

How to write high security smart contract [15] is the most important thing for developers, there are several ways as follows:

(1) follow the rules of contract development, such as avoiding external calls, trading order dependence.

(2) Custom secure contract templates, such as Open-Zepplin from Ethereum [16].

(3) Formal verification of smart contracts by using the formal verification platform of smart contracts [17], to avoid serious security loopholes.

The contract of warehouse receipt transaction in this article is mainly tested by Oyente tools [18]. The Oyente tool primarily detects the four most common security breaches in contracts, including reentrant attacks, transaction order dependencies, timestamp dependencies, misoperations. If there is no security issue in the contract, the returned test result shows that all four vulnerabilities correspond to a False status. If there is a certain type of security vulnerability in the contract, the vulnerable line of code is directly located so that the developer can quickly find and optimize the problem.

Reentrant attack problems have emeraged through the use of the Oyente tool to detect business contracts (such as User Contract, Market Contract, etc.). Through the analysis found that the main reason is the nested calls between contracts, such as user contracts in order to obtain market information and call the market contract, the current operation needs to be end after the call is completed. At this time, if the market contract needs to obtain user information, this creates a reentrant attack. In response to this problem, the event mechanism is used to avoid complex nested calls and to prevent reentrant attacks.

7. Summary and Prospect

This paper proposes the research and application of smart contracts based on OTC warehouse receipt transactions. The main work includes: PBFT consensus algorithm replaces the original consensus algorithm POW of Ethereum, modifies data structures such as transactions and blocks, and adds key management, node management, block cache mechanism, smart contracts design and so on. Finally achieved based on the blockchain smart contract warehouse receipt trading system.

At present, the application of smart contracts is still in its infancy, and smart contracts cannot support complex business scenarios. There are still some problems such as trustworthiness and security. The follow-up still needs further research on smart contracts. The specific ideas are as follows:

(1) adding zero knowledge proof [19] to enhance the privacy protection function of smart contract;

(2) study the fragmentation technology, off-chain code storage mechanism to solve the scalability of smart contracts;

(3) Tracking the latest smart contract security solutions and formal verification methods [20], verifying the consistency of smart contracts, developing smart contracts with higher security, and customizing more complex smart contract business templates.

This article is based on the blockchain smart contract technology in the OTC warehouse transaction business exploration and application. At present, the system's basic warehouse receipt transaction function has been completed and verified, which lays a solid foundation for developing more complex OTC business, and customizing business contract templates by using smart contract.

References

[1]. Nick S. Formalizing and securing relationships on public networks[J/OL]. First Monday. 1997[2017-12-20]. http://www.firstmonday.org/ojs/index.php/fm/article/view/548/469.

[2]. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009[2017-12-20], http://bitcoin.org/bitcoin.pdf.

[3]. Yuan yong, Wang Feiyue. Blockchain: The State of the Art and Future Trends [J]. Acta Automatica Sinica,2016, 42(4): 481-494.

[4]. Vitalik B. A next-generation smart contract and decentralized application platform[EB/OL]. 2014[2017-12-21]. https://github.com/ethereum/wiki/wiki/White-Paper.

[5]. Ethereum[EB/OL].2017[2017-12-21]. https://www.ethereum.org/.

[6]. Hyperledger[EB/OL].2017[2017-12-21]. https://www.hyperledger.org/.

[7]. RSK Labs. Rootstock: Smart contracts platform powered by Bitcoin.[EB/OL].2015[2017-12-21] http://www:rootstock:io.

[8]. Mike Hearn. Corda: A distributed ledger. [EB/OL].2016[2017-12-21]. https://docs.corda.net/_static/corda-technical-whitepaper.pdf.

[9]. Miguel Castro, Barbara Liskov. Practical Byzantine Fault Tolerance[J]. Proceedings of the Third Symposium on Operating Systems Design and Implementation.1999.

[10]. Zou Jun, Zhang Haining, Tang Yi. Blockchain Technical Guide [M]. Chain Machine Press. 2016.

[11]. Ethereum Foundation. The solidity contract-oriented programming language[EB/OL].2017 [2017-12-21]. https://github:com/ethereum/solidity.

[12]. Ethereum.Solidity-browser[EB/OL].2016[2017-12-21]. https://ethereum.github.io/browser-solidity.

[13]. Truffle[EB/OL].2017[2017-12-21]. http://truffleframework.com/docs/.

[14]. DAO Attack[EB/OL].2017[2017-12-21].https://www.coindesk.com/dao-attacked-code-issue-leads-60-million-ether-theft/.

[15]. Jack Pettersson, Robert Edstrom. Safer smart contracts through type-driven development[D]. Chalmers University of Technology and Gothenburg, Sweden. 2016.

[16]. Open-Zeppelin. Build Secure Smart Contracts in Solidity.2017[2017-01-02]. https://openzeppelin.org/.

[17]. Securify.Formal Verification of Ethereum Smart Contracts[EB/OL]. 2017[2017-01-02] https://securify.ch/.

[18]. Loi Luu, Duc-Hiep Chu, Hrishi Olickel, et al. Making Smart Contracts Smarter[J]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016:254-269.

[19]. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, et al. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge.https://eprint.iacr.org/2013/507.pdf.

[20]. Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet, et al. Short Paper: Formal Verification of Smart Contracts [J]. Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, 2016:91-96.

[21]. Hu kai, Bai Xiaomin, Gao Lingchao, et al. Formal Verification Method of Smart Contract[J]. Research on Information Security, 2016.