Research and Application of Smart Contract Based on Ethereum Blockchain

Yuxin Huang, Ben Wang* and Yinggui Wang
School of information science and engineering, Hangzhou Normal University, Hangzhou, 311100, China
616710866@qq.com, 20170056@hznu.edu.cn*, 2276598112@qq.com

Abstract. With the rapid development of blockchain technology and the increasing demand for partial decentralization of the Internet, the application of underlying technology based on blockchain has been widely concerned. Along with decentralized objects, the programmable financial system represented by Ethereum has been gotten more and more attention. However, smart contract sacrifices its security to improve decentralization. So Ethereum has the fatal problem with a large number of users, and negligence of users in coding contract threatens the entire Ethereum network. Therefore, this paper aims to research and expand applications of smart contract usage in Ethereum blockchain. We start from basic concepts to define structure of Ethereum, and then discuss security issues on smart contract. In the end, an optimized smart contract application of auction is implemented, which is useful for further consolidating and understanding of smart contract in practice.

1. Introduction
More than ten years development of Blockchain technology, the research topics have been changed considerably [1]. Technological innovation has gone through three stages: blockchain 1.0 mode, 2.0 mode, and 3.0 mode. Blockchain 1.0 mode is characterized by programmable digital encryption currency system, such as Bitcoin [2]. Blockchain 2.0 mode is characterized by programmable financial system such as Ethereum. And main feature of blockchain 3.0 mode is programmable society, which means blockchain participates are in various fields of society. In Figure 1, the arrows represent the parallel development of three phases. In fact, we are currently in blockchain 2.0 mode, but still need to improve and develop 1.0 and 2.0 mode when moving to 3.0 mode. The “smart contract” is in the core of Ethereum, and it actually has been implemented due to the emergence of blockchain. The smart contract itself, as a programmable application, has gradually been promoted the development of blockchain as well.
Looking at the overall history of blockchain, it has been more than a decade since Bitcoin was proposed in the late 2000s. A few years after Bitcoin became the first and the biggest case of blockchain technology, the emergence of Ethereum immediately changed the research direction on blockchain at that time [3]. Digital encryption currency moved into the programmable “era of smart contract”. Meanwhile, Ethereum is the most active public blockchain of 2.0 mode in the world at present. The unique feature of smart contract clearly distinguishes between Ethereum and Bitcoin. Ethereum is a collection of protocols that provide an Ethereum Virtual Machine (EVM)[4] that can execute any complex code and allow users to write their own applications [5]. Ethereum is a turing completeness system compared to complex script in Bitcoin. Thus, Ethereum offers more openness while drastically reducing the time for block generation and the amount of capacity of blocks. In terms of system block generation pattern, nowadays Ethereum is similar to Bitcoin, and it adopts the mining mode - Proof of Work (PoW) [1]. Furthermore, Ethereum designers set a “difficulty bomb”. It is the algorithm gradually reducing block reward of PoW after a certain period, and increasing block reward of Proof of Stake (PoS), so that the system could transform from PoS to PoW [6]. The preset consensus algorithm for Ethereum is defined as Casper, and the specific adjustment method will be mentioned with Ethereum structure in the second part.

Compared with most encryption currency, such as UTXO based on Bitcoin, the transactions themselves drive the state of blockchain system. Ethereum has proposed the concept of “account” with code that is more programmable and readable, and also suitable for batch transactions with smart contract. Account in Ethereum system is divided into external account and contract account, in which external account has a private key to access to Ethereum. The contract account has a contract code, but no private key. Therefore, only external account can start the transaction through private key. Contract account cannot initiate the transaction, but it can invoke other contracts to make the transaction request.

Ethereum has four stages: Frontier, Homestead, Metropolis, and Serenity. The Metropolis is divided into Byzantium and Constantinople. Ethereum once gets a callback of difficulty bomb and a reduction of block reward in Ethereum’s history during Byzantium stage of Metropolis. In addition, Ethereum has implemented Constantinople stage in early 2019 that allows smart contract to verify each other by hash values, and it also reduces block reward to two ETH while fixing security issues. The Constantinople hard fork was implemented in late 2019 to fix more security issues before the final stage of the transition. The Ethereum community planned to complete the final stages of transforming of consensus algorithm. But the release date for this “Serenity” stage is still to be determined as PoS is still in immature testing phase.

2. Data structure and consensus algorithm

2.1. Structure of Merkle Patricia Tree
In Ethereum, there are three types of trees in each block: transaction tree, receipt tree, and status tree. The transaction tree is quite similar to Merkle Tree in Bitcoin, except that the optimized Merkle
Patricia Tree (MPT) is used to store data. Both receipt tree and transaction tree store the transactions in the current block, and their respective nodes correspond to each other. As the execution process of smart contract is relatively complex, receipt tree is convenient for the rapid query of data in system. State tree stores the state of all accounts in Ethereum system, which is the proposed data structure for Ethereum account concept. All three kinds of trees in Ethereum adopt MPT structure, and it is the same as Merkle by storing key-value pair. In status tree, the key of account state is 40-bit hexadecimal address, and the value of account state is the balance of user. The advantages of MPT used in Ethereum system are as follows:

- The number of branches of MPT in Ethereum system is determined by the range of each element of key, which is 17 bits hexadecimal 0-f plus an ending identifier.
- The query efficiency of MPT in Ethereum system depends on the length of key, which is 40 bits length of account address.
- Hash collisions will not occur while it is possible in Bitcoin system, because different account addresses must be on different branches in MPT.
- Regardless of input order, the result ultimately processed by MPT must be the same for any key-value pairs with the same content.
- MPT provides Ethereum with a local update operation. As long as a small part of the account information changes in system, most of the unchanged tree node data share tree node data in the previous block. It avoids the time cost by re-traversing account and reconstructing state tree for each block.
- MPT is based on the original Trie structure through path compression, greatly saving the storage space of tree, thus speeding up the data query.

2.2. Rules of block generation
Comparing with block generation in Bitcoin with 10 minutes on average, Ethereum reduces to 15 seconds in average. It increases the possibility of blockchain fork. This kind of temporary soft fork will cause certain unfairness. Therefore, Ethereum utilizes “ghost protocol”, and introduces a concept of “uncle block” - nodes get right to charge an account ledger, but may not be the longest chain. Also, this legal temporary fork block has a common “ancestor” block with the latest blocks in seven generations. And the new node for charging an account ledger can decide whether the previous block is uncle block or not.

As shown in Figure 2, the system gives uncle block a certain amount reward. For example, the previous generation uncle block in new block gets 7/8 of block reward, and the previous of it gets 6/8, until uncle block of the seven generation gets 2/8. The new node can get extra reward if it contains the uncle block at the same time, each containing can rise 1/32 block reward.

2.3. Consensus algorithm
In terms of the consensus algorithm, Ethereum plans to use PoW at the beginning and convert it to PoS at the later stage. It is embodied in the difficulty bomb $\epsilon$ in the following mining difficulty
formula. The $D(H_i)$ in Mining difficulty formula (1) is the difficulty of this block, which is adjusted on the basis of difficulty of the parent block $P(H_{i,H_d})$; $D_0$ is the minimum value of the lower difficulty limit of 131072; $x \times \sigma_2$ is used to adjust the difficulty of block adaptively according to time of block generation, so that the speed of block generation remains unchanged; Difficulty bomb $\varepsilon$ is used for the transition to PoS.

$$
D(H_i) = \begin{cases} 
D_0, & H_i = 0 \\
\max(D_0, P(H_{i,H_d}) + x \times \sigma_2) + \varepsilon, & \text{others}
\end{cases} \quad (1)
$$

The difficulty adjustment unit $x$ in $x \times \sigma_2$ is defined in formula (2), and 1/2048 of the difficulty of parent block is taken as a unit of difficulty adjustment.

$$
x = \left| \frac{P(H_{i,H_d})}{2048} \right| \quad (2)
$$

The difficulty adjustment coefficient $\sigma_2$ in $x \times \sigma_2$ is defined in formula (3), where $y$ is the number of uncle block of parent block plus 1, and $H_i$ is the timestamp of current block. In the formula, $H_{s-H_i}$ is a time interval between the current block and the previous block, then give 1/9 of the interval adjusted. For example, if the time interval is less than 9 seconds with no uncle block, $\sigma_2$ is 1, which means that the difficulty will be increased by one unit. If the time interval is larger than 18 seconds with no uncle block, $\sigma_2$ is -1, which means that the difficulty will be reduced by one unit. A minimum value of -99 is set for $\sigma_2$ to prevent the difficulty from changing rapidly due to malicious attacks.

$$
\sigma_2 = \max\left(y - \left| \frac{H_{s-H_i} - P(H_{i,H_d})}{9} \right| - 99 \right) \quad (3)
$$

The purpose of the difficulty bomb $\varepsilon$ is to avoid hard fork in Ethereum community due to the transforming of consensus algorithm. In the formula of difficulty bomb (4), $H_i$ represents the total number of blocks in current Ethereum. It can be seen from formula that when the total number of blocks is larger than 300000, the difficulty bomb will start to show power. Then, the difficulty bomb will double for every 100000 blocks, and the mining difficulty will also increase exponentially. Finally, Ethereum community users were forced to give up mining from PoW to PoS with less block reward and more PoS reward.

$$
\varepsilon = \left| 2^\left\lfloor H_i + 100000 \right\rfloor - 2 \right| \quad (4)
$$

But for now, PoS has not yet reached a level where can be safely used in Ethereum. When difficulty bomb is gradually increasing the power, Ethereum community modifies the variables (5) in difficulty bomb for continuing to work in PoW: $H_i'$ are false block number, means that it is a number 3 million less than it actually is.

$$
H_i' = \max(H_i - 3000000, 0) \quad (5)
$$

3. Smart contract

Nowadays there are several smart contract projects in the world, such as Ethereum smart contract platform, Rootstock smart contract platform [7] and Hyperledger project [8]. The introduction to current smart contract projects is shown in Table 1. One of the most important projects, Ethereum smart contract, is focused in this paper.

| Smart contract | Research | Introducing to projects |
|----------------|----------|------------------------|
|                |          |                        |
| projects       | institution                      |
|----------------|----------------------------------|
| Ethereum       | ethereum.org                     |
| Rootstock      | RSK                              |
| Hyperledger    | IBM                              |

3.1. structural logic

On the basis of decentralization of digital currency by Bitcoin, Ethereum considered whether it could also decentralize other things besides digital currency. As a result, smart contract for decentralized application was born. The smart contract is essentially a piece of code running on blockchain. Different from the entity contract, the content of contract is defined by code logic, so people often say “code is law”. The smart contract code automatically triggers contract that is managed by a central organization in the past, and the code is placed on blockchain with immutable characteristics. Hence, the unique smart contract model of Ethereum makes it a popular blockchain application after Bitcoin. The current running state of smart contract is stored in contract account, which contains current balance, nonce, code, and MPT. The contract account creates a smart contract and publishes it on Ethereum via a special transaction with a destination address 0x0. The contract is stored in blockchain as an address through which an external account can invoke contract. The contract itself is only executed when it is called by a transaction. The contract account has no additional privileges comparing with other account, so it is generally referred to as “smart contract”.

A high-level programming language for smart contract is Solidity, similar to JavaScript. The function in contract can be called by other contract. In this case, there are several ways to call contract. Firstly, the function is called directly to the function. Once the function throws an exception, the starting function also throws an exception, and then all the exceptions are rolled back. Secondly, user invokes addr.call() function. It will not cause the starting function to throw an exception if the called function have an error. Thirdly, user invokes delegatecall(). The advantage is that the code storage in another contract can be used.

The smart contract is similar to a traditional contract, with a three-step process of contract generation, release, and execution. The specific process is shown in Figure 3. The contract generation stage mainly includes negotiation of participants early, discussion on how to reach consensus, design of contract specifications, and write contract code. The code of smart contract requires strict discussion and verification of participants. The second stage is contract release stage, in which tests and verification are conducted before the contract is released. Once smart contract code is deployed in blockchain, it cannot be modified again. The contract is distributed to each node as a transaction, and then it is added to blockchain. At last, contract execution stage is processed. External account is required to invoke the contract in Ethereum. When the condition is met, smart contract will be executed immediately. Thus, the automatic characteristics of smart contract are different from traditional contract.
3.2. Example of application
Smart contract on Ethereum blockchain has been discussed in detail. In Figure 4, a basic auction application is implemented by smart contract. The code is implemented by Solidity and tested it on the Ethereum private chain [9]. Generally, auction in the contract has the same logic as regular auction, but all operations are presented in decentralized form. As long as the auction conforms to certain rules, it can be carried out successfully until it is completed. In the auction process, all bidding is conducted through the payment of Ether. Nodes with previous payment records only need to send the bidding difference. Finally, the bidder with the highest auction price gets the auction item, and the highest price transfers to beneficiary. And then any other nodes, the loser of the auction, invoke the callback function to roll back cost in the auction.

Function: first basic auction function application

```
1 Begin
2 build an auction{
3   ‘biddingTime’ for auction time
4   ‘beneficiary’ for address of beneficiary
5   participant bids on the auction{
6     if auction is not over
7       if previous bid amount plus this bid is higher than the highest bid
8         if the user has not bid before, joins the list of bidders
9         the information of the highest bidder is logged
10       the highest bid is sent to beneficiary, the remaining bid is returned{
11         if auction is over
12           if the function is not called
13             the highest bid is sent to beneficiary
14             call callback function, get others Ether back
15     End
16   End
17 End
```

Figure 4. Flawed auction function application with smart contract.

Due to the potential and unavoidable risk of callback function in this contract, user may call it through wrong contract account, which leads to continuous rollback of function once it encounters the wrong process. Finally, it returns to the state before the call, and the auction balance is locked. Therefore, the modified smart contract for all participants get their own bid back, and divided the last function into two parts: sending to beneficiary and fallback function. However, it is difficult to avoid
the Re-entrancy Attack to retrieve the amount repeatedly, so contract code will be modified until the final optimized version is written in Figure 5. This version of smart contract is preferred on Ethereum chain for the secure auction activities.

| Function: second basic auction function application |
|---------------------------------------------------|
| 1 Begin                                           |
| 2 build an auction{                               |
| 3 ‘biddingTime’ for auction time                  |
| 4 ‘beneficiary’ for address of beneficiary}       |
| 5 participant bids on the auction{                |
| 6 if auction is not over                           |
| 7 if previous bid amount plus this send bid is higher than the highest bid |
| 8 if the user has not bid before, joins the list of bidders |
| 9 the information of the highest bidder is logged} |
| 10 bidders retrieve bid themselves{               |
| 11 if auction is over                              |
| 12 if be the highest bidder, can not retrieve     |
| 13 if the current address has balance             |
| 14 balance of this bidder in hash table set 0      |
| 15 balance is transferred to caller}              |
| 16 the highest bid is sent to beneficiary{        |
| 17 if auction is over                              |
| 18 if the highest bid larger than 0                |
| 19 balance of beneficiary in hash table set 0      |
| 20 the highest bid is sent to beneficiary}        |
| 21 End                                            |

Figure 5. Improved auction function application with smart contract.

4. Conclusion
In this paper, the structure of Ethereum and the operation logic of smart contract are analyzed and improved to handle security issues of smart contract. Therefore, the concept and theory of Ethereum are presented in blockchain application, with the data structure of Ethereum. Then the operation mechanism of smart contract is presented with pseudo code at the technical level for smart contract, which is the core of Ethereum. In the real experiments of smart contract, the auction is implemented by Solidity language. At last, the defects of the auction application are fixed and improved to support the application of smart contract in Ethereum blockchain.

Acknowledgments
This paper is funded by projects: Basic public welfare research program of Zhejiang Province (lgf19F020011) [10]; Independent application project of agricultural and social development scientific research in Hangzhou (20191203b14) [11]; General scientific research project of Zhejiang Provincial Education Department in 2020 (Y202044936).

References
[1] Nakamoto, S. (2008) Bitcoin: a peer-to-peer electronic cash system. https://git.dhimel.com/bitcoin-whitepaper/.
[2] Swan, M. (2015) Blockchain: Blueprint for a new economy. O’ Reilly Media Inc. Sevastopol.
[3] Buterin, V. (2014) A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper.
[4] Delmolino, K., Arnett, M., Kosba, A. (2016) Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In: International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg. pp. 79-94.
[5] Zhang, S., Yang, Y.-G. (2018) The Foundation and Application of Blockchain Technology. Journal of Information Security Research, 4: 575-580.

[6] King, S., Nadal, S. (2016) PPcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. https://decred.org/research/king2012.pdf.

[7] Lerner, SD. (2015) RSK White paper overview. https://www.chainwhy.com/upload/default/20181122/3391764f0388d08d406494fa1a8e.pdf.

[8] Hyperledger Performance and Scale Working Group. (2018) Hyperledger Blockchain Performance Metrics. https://www.hyperledger.org/learn/publications/blockchain-performance-metrics#.

[9] Huang, Y.-X., Wang, B., Wang, Y.-G. (2020) MResearch on Ethereum Private Blockchain Multi-nodes Platform. In: 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). Fuzhou. pp. 369-372.

[10] Wang, B., Zhou, W.-L. Comprehensive Integrated Platform for Garbage Classification in Reduction, Innocuity, and Resource[J]. Computer, Communication and Network Technology. 2018. pp. 26-31.

[11] Wang, B., Zhou, W.-L., Li Z.-H. Research on Distributed Intelligent Mattress on the Internet of Things[J]. Computational Modeling[J], Simulation and Applied Mathematics. 2018. pp. 323-327.