Formal Modeling and Verification of Blockchain System

Zhangbo DUAN
State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
(86)18610613100
duanzhangbo@buaa.edu.cn

Hongliang MAO
National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 110105, China
mhl@cert.org.cn

Zhidong CHEN
State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
(86)18500226278
czdbuaa@buaa.edu.cn

Xiaomin BAI
State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
(86)13051386669
baixiaomin@buaa.edu.cn

Kai HU
Corresponding author
State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
(86)010 82339460
hukai@buaa.edu.cn

Jean-Pierre Talpin
Institut National de Recherche en Informatique et en Automatique (INRIA) Rennes, Rennes, France
(33) 299847436
jean-pierre.talpin@inria.fr

ABSTRACT
As a decentralized and distributed secure storage technology, the notion of blockchain is now widely used for electronic trading in finance, for issuing digital certificates, for copyrights management, and for many other security-critical applications. With applications in so many domains with high-assurance requirements, the formalization and verification of safety and security properties of blockchain becomes essential, and the aim of the present paper. We present the model-based formalization, simulation and verification of a blockchain protocol by using the SDL formalism of Telelogic Tau. We consider the hierarchical and modular SDL model of the blockchain protocol and exercise a methodology to formally simulate and verify it. This way, we show how to effectively increase the security and safety of blockchain in order to meet high assurance requirements demanded by its application domains. Our work also provides effective support for assessing different network consensus algorithms, which are key components in blockchain protocols, as well as on the topology of blockchain networks. In conclusion, our approach contributes to setting up a verification methodology for future blockchain standards in digital trading.

CCS Concepts
• **Software and its engineering→Software organization and properties→Software functional properties→Formal methods→Model checking.**

Keywords
Blockchain protocols; Formal methods; Model-checking; Formal Verification.

ICCMS 2018, January 8–10, 2018, Sydney, Australia

ACM ISBN 978-1-4503-6339-6/18/01…$15.00
https://doi.org/10.1145/3177457.3177485

1. INTRODUCTION
The notion of blockchain [1] is an emerging decentralized protocol, network architecture and distributed computing paradigm, which constitutes the core technology of the digitally encrypted monetary system implemented by Bitcoin [2]. The notion of blockchain [1] is an emerging decentralized protocol, network architecture and distributed computing paradigm, which constitutes the core technology of the digitally encrypted monetary system implemented by Bitcoin [2]. It performs peer-to-peer transactions, coordination and collaboration in a distributed and decentralized network by means of data encryption, time stamping, distributed consensus and economic incentives. However, with the application in so many areas with high-assurance requirements, the study of security and safety of blockchain becomes particularly important. To make blockchain safe and reliable platforms for information sharing, and the praised solution for value transfer, formal methods [3] must be employed to provide sufficient mathematical evidences of security and safety guarantees to users.

Formal methods [3] are mathematically based techniques for the specification, development and verification of software and hardware systems that provide irrefutable evidence as to the reliability and robustness of a system design, up to its specification. Formal verification employs two types of methods: model checking and theorem proving. They are of great significance to the development of embedded real-time systems [4], network protocols [5], credit card authentication systems or other High-Assurance systems.

The blockchain network can be regarded as an instance high-confidence distributed communication protocol. Therefore, the introduction of formal methods can effectively improve the security and overall credibility of the blockchain system to end users. We introduce a hierarchical and modular modeling method to formalize the blockchain protocol. This model contributes to improving the development automation level, shortening the development time, and reducing the possibility of error in the coding process. Traditional blockchain experiments often require the deployment of a large number of computers to meet their experimental conditions. With the blockchain formal modeling method, multi-node blockchain simulation can be achieved on a single computer. Furthermore, it allows us to study the properties

of different consensus algorithms, network topologies and inter-blockchain transactions using smart contracts [6] over a given blockchain network by solely relying on our blockchain model. This all significantly contributes to developing, experimenting and assessing solutions for the blockchain standards of the future. In summary, the main contributions of this paper are:

1) We introduce the formal modeling and verification method for blockchain systems;
2) We design a hierarchical and modular SDL model of a crowdfunding private blockchain [7] using Telelogic Tau [8];
3) We study simulation and formal verification method of the blockchain model.

The paper is organized as follows. Section 2 reviews related works. Section 3 presents our SDL model of a crowdfunding private blockchain in Telelogic Tau. Section 4 presents the simulation and verification of that blockchain model. Section 5 concludes our work.

2. RELATED WORK

2.1 Formal Modeling Language

A critical issue of this paper is to find an appropriate formal modeling language to model the blockchain system. Since a blockchain system can be abstracted as a network protocol, we have considered several types of model checking formal languages based on finite state machines, which are listed below.

SDL [9] (Specification and Description Language) is an object-oriented, formal language defined by the International Telecommunication Union (ITU) based on EFSM (Extended Finite State Machine). SDL is able to describe the structure, behavior, and data of real-time and distributed communicating systems with a mathematical rigor that eliminates ambiguities and guarantees system integrity. It is widely used in network protocol engineering. IEEE has begun to use SDL to define its standards.

ESTELLE [10] is also based on EFSM, but it uses Pascal syntax and data types. The EFSM of ESTELLE is basically the same as SDL, but they are different in some concepts. ESTELLE is mainly used for distributed, parallel message processing systems, communication protocols and services.

PROMELA [11] (Process or Protocol Meta Language) is a verification modeling language introduced by Gerard J. Holzmann. The language allows for the dynamic creation of concurrent processes to model, for example, distributed systems.

Among those formal languages, SDL has very strong description, verification and simulation capabilities. Along with its tool Telelogic Tau, it is widely used in industry. Therefore, we will use SDL as our blockchain modeling language.

2.2 The Application of Formal Methods in Blockchain

In the field of blockchain, application of formal methods has been proposed several times, but available implementations are still few.

[12] proposes a temporal "rolling" blockchain and proof that this rolling blockchain maintains the key security principles and provides the same security properties as a traditional blockchain. It does not consider introducing any additional vulnerability. The approach uses the B language to model the blockchain system. The model is simple and intuitive but its implementation is closed source.

In [13], the author modeled Bitcoin blockchain using Petri nets and analyzed the connections between places (bitcoin addresses) and transitions (bitcoin transactions). Again, the model does not scale to a suitable level of detail to fully specify the implementation of all protocols involved.

Another important application of formal methods concerns smart contracts executed on blockchain. [14] uses F* language to verify smart contracts. This paper shows that the type and effect system of F* is flexible enough to express and prove non-trivial properties.

Therefore, the SDL modeling, simulation and verification method based on blockchain proposed in this paper effectively fills the gaps of related proposals. The hierarchical modular SDL modeling method provides effective support for further developments, such as consensus algorithms, network topology and inter-blockchain transactions.

3. MODELING

3.1 Hierarchical Modeling

For a blockchain system, we can divide it into five layers as shown in Figure 1 : application layer, smart contract layer, consensus layer, network layer and data layer.

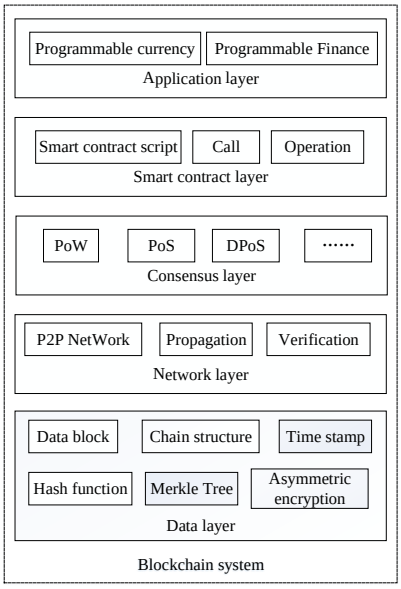

Figure 1. Blockchain infrastructure.

The data layer is for block generation, blockchain construction and storage. The network layer has functions such as unicast, broadcast, multicast, and filtration of data packets. The ability to join and delete nodes is also necessary. Consensus layer is the core level of the blockchain. It defines the rules of the blockchain operation, such as agreement of nodes, backup, fault tolerance, data consistency and partition fault tolerance. Smart contract layer implements the write, compilation and execution of smart contract. The application layer is the upper layer of various applications, such as electronic money system of the first generation blockchain, programmable finance, medical and supply chain of the second generation blockchain. Among these layers, data layer, network layer and consensus layer implement the main functions of blockchain system.

In SDL, a structure can be described at three levels [15]: system, block, or process. A system represents the object to be described. A system communicates with the outside environment through channels and signals. A system contains one or several blocks and a block contains one or several processes. Most functions of a SDL system are described the process level. SDL relates to conventional programming languages by using blocks to represent modules and processes to describe functionalities.

Based on the analysis of blockchain characteristics and the organization of SDL, we propose a blockchain-based hierarchical modeling method. In this paper, we focus on the most important component of blockchain: the consensus layer.

3.1.1 Analysis and the definition of data structure

As distributed and decentralized architectures, blockchain have not yet gained the status international standards. We choose a private blockchain system introduced in [7] as our modeling target. In this system, an Improved Byzantine Fault Tolerant Algorithm is applied as consensus algorithm.

First, we define signals and data structures in our model. A block in SDL is presented as a pair with one header BT and a body BH:

$B \equiv (B_H, B_T)$ with $B_H \equiv \langle \mathbb{H}_p, \mathbb{H}_a, \mathbb{H}_t, \mathbb{H}_m, \mathbb{H}_h \rangle$ and $B_T \equiv \langle T_1, T_2, \dots, T_{\mathbb{H}_l} \rangle$

In the header, $\mathbb{H}_p$represents the hash value of previous block, $\mathbb{H}_a$ the hash value of this block, $\mathbb{H}_t$ the timestamp, $\mathbb{H}_m$ the root of merkle tree, and $\mathbb{H}_h$ current block height. The essence of the blockchain system is a distributed database, and the data stored in the database are collectively referred as "transaction" in the blockchain system. In our model, due to the performance of the SDL tool, we use an integer type to represent a transaction. Figure 2 shows the definition of a block structure in our model.

```
newtype blockchain struct
  prehash integer;
  hash integer;
  length integer;/*the length of blocks*/
  merkleroot integer;
  ti Time;/*Timestamp*/
  translist list;
endnewtype;

newtype list
  array(maxit, integer)
endnewtype;

syntype maxit =Integer constants 0:25
endsyntype;
```

Figure 2. Definition of a block structure.

As a list of data blocks linked in chronological order, blockchain can be regarded as a block-driven singleton state machine, including the non-empty state set, the input transaction set, the state transition function, the start state, and the acceptance state set. The formal description of the blockchain states is as follows:

$$\mathcal{S} \equiv (Q, \Sigma, \delta, s, F)$$

Among them:

- Q is a non-empty state set, which is all states of the blockchain system.
- Σ is the set of new generated and consensus blocks.
- δ is the state transition function, $\delta: Q \times \Sigma \to \mathrm{Q}$. For example, $\delta(\sigma, B) = \sigma'$, in which $\sigma, \sigma' \in Q$, $B \in \Sigma$.
- s is the start state, which is the state of the global system when the system is initialized. $s \in Q$
- F is the acceptance state set. $F \subseteq Q$.

At the beginning of the model, a genesis block is generated. At this time, the state of the model is the start state. As the transactions arrive, the model begins to package the transactions and selects the leader node to generate a block. The block is broadcast by the leader node to other nodes in the blockchain network. Other nodes receive and verify the block to determine whether the block passes consensus. The consensus block stores the hash of the previous block so as to add it at the end of the blockchain and complete the transfer of the blockchain state.

3.1.2 Definition of system and block

According to the above analysis and SDL features, the system diagram and the block diagram of single node in the blockchain system are shown in Figure 3.

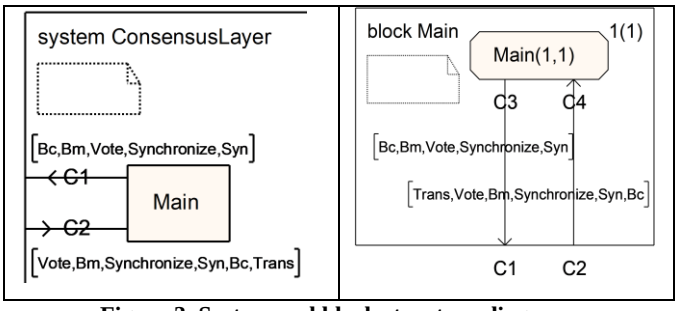

Figure 3. System and block structure diagram.

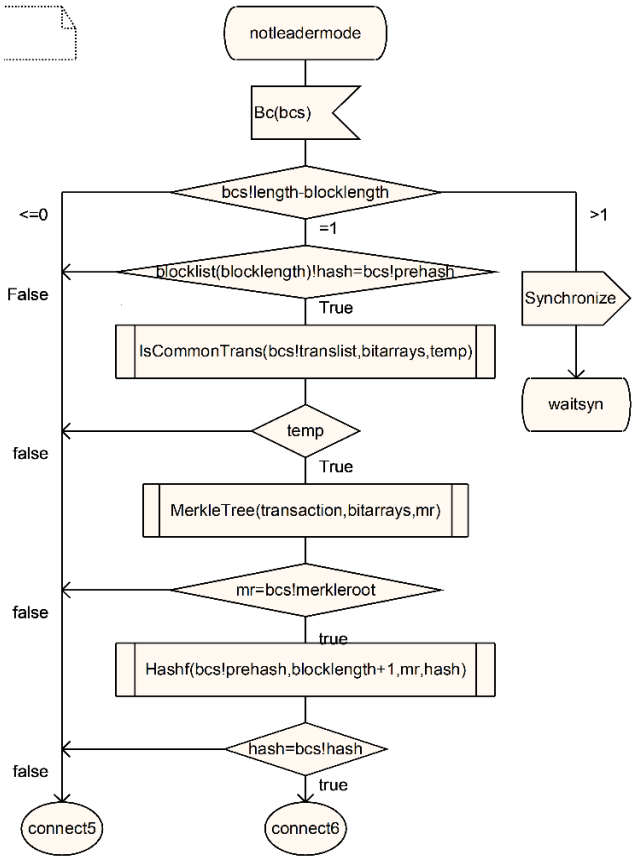

Figure 4. Example of process.

3.1.3 Definition of process

Inside the process Main, 14 process diagrams and 6 procedures implement the functions of a single node in the blockchain system. An example of process is shown in Figure 4. This part shows the process of block verification after the current node receives the block generated by the leader node.

Firstly, we compare the size of the current node with the received block size. If the height of the current node is greater or equal than that of the received block, the latter is considered wrong or resent and voted against. If the size of the current node is less than the height of the received block by one, this means that the block of the current node is missing. Hence, we perform a block synchronization. Finally, if the current node size equals the received block size minus one, then the block size is valid. In the next step, we verify that the hash of the previous block equals the prehash present in the header of the cur-rent block. If not, it we vote against it. Finally, we verify the transactions, merkle tree and hash value. Once all the verifications are passed, we vote in favor of the block.

3.2 Modular Modeling

A blockchain network connects multiple blockchain nodes through a specific topology. As the case of reaching consensus of Byzantine Failures is $n_g > 3n_b$, in which n_g means the number of good nodes and n_b means the number of bad nodes, so the minimum number of nodes in a private blockchain network is four.

According to the previous blockchain consensus layer model, we can establish a four-node blockchain network model as shown in Figure 5. It is a model with 4 nodes Node1 to Node4. Every node is a SDL block which contains a single node as described in 3.1.2.

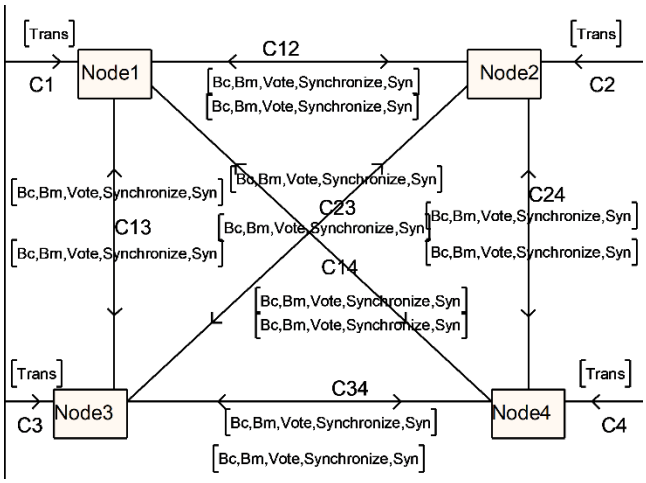

Figure 5. System structure diagram.

A modular blockchain network model is of great significance to conduct further research such as consensus algorithms, network topology or inter-blockchain transaction. It can further simplify the implementation of the software and hardware environment of a traditional blockchain network by, e.g., allowing to generate modules of code from models, automatically. Finally, the generic model can be applied to additional case studies and applications of blockchains, hence reducing the design workload considerably.

4. SIMULATION AND VERIFICATION

4.1 Simulation

The simulation the blockchain network allows us to check whether all of its functions are operational. Before the simulation, a static analysis should be done to guarantee the correctness of syntax and semantic of the model. For this part, we use the simulator module of Telelogic Tau. In our model, we add a TransSend block, which sends 25 virtual transactions (every transaction is presented by a random number) to 4 nodes every minute. We only have to input a "Begin" signal to start the whole system. To ensure the validity of our Improved Byzantine Fault Tolerant Algorithm, we assume that Node1 is a bad node (which means Node1 always receive different transactions with other 3 nods) and restart our simulation.

We can observe a running fragment of the blockchain simulation. In the following we have an example in Figure 6.

First, we observe the blockchain storage status of each node. We can see that every node has exactly the same storage status. Their blockchain length are all 7. (Transactions are so numerous that only part of the blockchain storage is shown below. We carefully compared the blockchain storage of these 4 nodes. The results show that they are all exactly the same. Besides, all these 7 blocks plus the genesis block are linked in chronological order by hash value).

```
Command : Main1 blocklength
blocklength (integer) = 7

Command : Examine-Variable ( Main2 blocklength
blocklength (integer) = 7

Command : Examine-Variable ( Main3 blocklength
blocklength (integer) = 7

Command : Examine-Variable ( Main4 blocklength
blocklength (integer) = 7
```

```
xamine-Variable ( Main1 blocklist
blocklist (chain) = (: (others:(. 0, 0, 0, 0, 0.0000,
8281, 53, 26418, 6900, 18127, 3728, 24648, 17807, 143

Command : Examine-Variable ( Main2 blocklist
blocklist (chain) = (: (others:(. 0, 0, 0, 0, 0.0000,
8281, 53, 26418, 6900, 18127, 3728, 24648, 17807, 143

Command : Examine-Variable ( Main3 blocklist
blocklist (chain) = (: (others:(. 0, 0, 0, 0, 0.0000,
8281, 53, 26418, 6900, 18127, 3728, 24648, 17807, 143

Command : Examine-Variable ( Main4 blocklist
blocklist (chain) = (: (others:(. 0, 0, 0, 0, 0.0000,
8281, 53, 26418, 6900, 18127, 3728, 24648, 17807, 143
```

Figure 6. Simulation result.

The simulation of Telelogic Tau can also generate MSC charts which clearly demonstrate the proper, deterministic, operations of the system including the input and output of signals, state transition, and whether or not the timer has overflowed. For reasons of space, we will not go into details here.

4.2 Verification

After modeling and simulation of the blockchain system, the static analyzer can find most of the static lexical and typing errors. However, it cannot find dynamic runtime errors such as deadlock, livelock, boundedness or state ambiguities, unreachable states. To this end, we use the verification tool of Telelogic Tau: Validator.

When Validator is used to verify an SDL model, the whole SDL model is replaced by a behavior tree. In the behavior tree, one node represents one state of the model. The set of all states forms the state space. Since SDL is a model-checking formal language, the verification of SDL consists of the exploration and traversal of the state space. Telelogic Tau provides 6 methods of traversal: Bit-State, Random Walk, Tree-Walk, Tree-Search, Exhaustive,

Verify-MSC. The algorithm of Bit-State exploration is commonly used for large models. It uses a hash table to reduce the space during exploration. Figure 7 shows our verification result. From Figure 7, we can draw the following conclusions:

1) Number of reports

There are 12 reports which are all about warnings of a deadlock (These 12 warnings are all the same except for their sender and receiver). We can click on this report to check out the detailed error trace.

From the MSC validator trace, we can extract the sequence of actions leading to the deadlock: if only one node can normally receive transactions while the other three cannot, the entire blockchain network will fall into a deadlock. This is what we did not take into account when we initially developed the crowdfunding private blockchain [7] system.

```
** Starting bit state exploration **
Search depth    : 100
Hash table size : 1000000 bytes

** Bit state exploration statistics **
No of reports: 12.
Generated states: 85.
Truncated paths: 0.
Unique system states: 45.
Size of hash table: 8000000 (1000000 bytes)
No of bits set in hash table: 90
Collision risk: 0 %
Max depth: 6
Current depth: -1
Min state size: 2628
Max state size: 2864
Symbol coverage :  17.33
```

Figure 7. Verification result.

Except from these warnings, there are no other reports such as livelock, boundedness or state ambiguities. After we corrected this deadlock error, we re-simulated and validated the entire model and found no other errors.

2) Collision risk

The rate of collision risk is 0%. This shows the performance of the verification is good and the verification result is more reliable.

3) Symbol coverage

The rate of symbol coverage is 17.50%. Although it is far from 100%, it is not a bad result considering that some process cannot be explored because the timer signals can-not be sent from the validator.

Through the above analysis, we have optimized our blockchain model and have made the corresponding improvements to our source implementation. The benefits of formal verification for blockchain development has hence been fully demonstrated.

5. CONCLUSION

The blockchain system is a system that requires a high degree of security and robustness, safety. To this end, we have introduced a formal modeling and verification methodology for the development of blockchains. In this paper, we have proposed a hierarchical and modular modeling method using SDL. The simulation and verification of the blockchain model has been demonstrated to provide significant benefits to the safety and security of development.

The application of formal methods can effectively in-crease the security and robustness of blockchain systems, improve the development automation level and shorten the development time. It can also help to define blockchain standards. Based on the above results, our next step will be to study different consensus algorithms for blockchains using SDL, study their performance, and verify them for reachability, deadlock, livelock, boundedness and state ambiguities.

6. ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under Grant 61672074 and 91538202, Funding of Ministry of Education and China Mobile MCM20160203, Project of the State Key Laboratory of Software Development Environment of China under Grant SKLSDE-2016ZX-16

7. REFERENCES

[1] Wright A, De Filippi P. Decentralized blockchain technology and the rise of lex cryptographia[J]. 2015.

[2] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Consulted, 2009.

[3] Clarke E M, Wing J M. Formal methods: State of the art and future directions[J]. ACM Computing Surveys (CSUR), 1996, 28(4): 626-643.

[4] Yang Z, Hu K, Ma D, et al. From AADL to Timed Abstract State Machines: A verified model transformation[J]. Journal of Systems & Software, 2014, 93(2):42-68.

[5] Hu K, Liu C, Liu K. Modeling and verification of custom TCP using SDL[C]// IEEE International Conference on Software Engineering and Service Science. IEEE, 2013:455-458.

[6] English S M, Orlandi F, Auer S. Disintermediation of Inter-blockchain Transactions[J]. arXiv preprint arXiv:1609.02598, 2016.

[7] Chen Z. Research on Private blockchain Based on Crowdfunding[J]. Journal of Information Security Research, 2017, 3(3): 227-236.

[8] Telelogic B. Telelogic Tau modeling tool[J]. 2010.

[9] Abed S, Al Shayeji M H, Ahmed O, et al. Formal Specification and Description Language and Message Sequence Chart to Model and Validate Session Initiation Protocol Services[J]. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 2016, 10(3): 512-520.

[10] Dis B I. Estelle, a formal description technique based on an extended state transition model[J]. ISO, 1988.

[11] Mikk E, Lakhnech Y, Siegel M, et al. Implementing statecharts in PROMELA/SPIN[C]//Industrial Strength Formal Specification Techniques, 1998. Proceedings. 2nd IEEE Workshop on. IEEE, 1998: 90-101.

[12] Dennis R, Owenson G, Aziz B. A temporal blockchain: a formal analysis[C]//Collaboration Technologies and Systems (CTS), 2016 International Conference on. IEEE, 2016: 430-437.

[13] Pinna A. A Petri net-based model for investigating disposable addresses in Bitcoin system[J].

[14] Bhargavan K, Delignat-Lavaud A, Fournet C, et al. Short Paper: Formal Verification of Smart Contracts[J].

[15] Lifa Wu, “Network Protocol Engineering”, Beijing, China: Publishing House of Electronics Industry,2011,77-79.