Threat Analysis Method and Smart Contract Verification to Improve Reliability of Blockchain

● Fumihiko Kozakura ● Shingo Fujimoto ● Yoshihide Nomura
● Kazuhiro Yamashita

Blockchain is a technology devised as the fundamental technology of Bitcoin. It allows reliability to be ensured in a decentralized manner, and expectations are high for its application in various areas beyond virtual currency, such as real estate and healthcare. However, any problem in the boundaries between the elements that constitute an entire system or in the smart contracts that are often introduced as subsystems may lead directly to significant losses in business, such as the theft of the virtual currency managed by a blockchain. Accordingly, improving the reliability of the system as a whole, including applications, is needed. Fujitsu Laboratories has developed a threat analysis method that checks a blockchain system for any problems at the time of its construction and operation, and a smart contract verification technology that exhaustively detects threats in smart contracts by using static analysis technology. These will enable blockchain developers to quickly develop systems that use blockchains with greater security. This paper describes the threat analysis method and smart contract verification technology that we have developed.

1. Introduction

Blockchain was devised as the fundamental technology of Bitcoin. This technology uses hash values and electronic signatures to link together blocks of transactions like a chain. As a result, the transaction history is not easily altered even in a non-centralized environment, realizing a high level of reliability ideal for transactions such as virtual currency transactions that require strong reliability. This technology is also used in highly reliable digital asset management systems that apply electronic signatures, and expectations are high for its application in various areas besides finance, such as real estate and healthcare.

Blockchain has withstood a variety of attacks because all transaction records are shared through a simple fault-tolerant peer-to-peer (P2P) network. However, it only has functions for the safe management of transaction information, making it ill-equipped for application to fields other than finance. For this reason, it is often used via a subsystem such as smart contracts, which are programs that automatically process transactions defined in the blockchain according to strict rules or a web front system. On the other hand, at the boundary with the subsystem, there may be a tradeoff between the improved convenience offered by system usage and new threats leading to double execution or spoofing. Thus faulty system design opens the door to potentially serious business loss.

Blockchain is broadly classified into three types according to its operation: public, consortium, and private. The differences among the three types are explained by the role of administrators. A public blockchain has no administrators, and the security of the system is assured by the mining process, in which miners compete to calculate hash values. By contrast, consortium and private blockchains have administrators that allow participants to construct and operate the system. The consortium is premised for use by select organizations that are authorized by administrators and others.

Fujitsu Laboratories has developed a threat analysis method that checks the blockchain system for any problems, and a smart contract verification technology that exhaustively detects threats in smart contracts by using static analysis technology. These technologies use Hyperledger Fabric (hereafter, Fabric),[1)] an execution platform for consortium and private blockchain. These technologies will enable blockchain developers to quickly develop systems that use blockchains with greater security.

This paper describes the threat analysis method and smart contract verification technology that we have developed.

2. Issues in building a blockchain system

This section explains two major categories of issues in building a blockchain system. The first one is issues related to the design of the boundaries between the elements that make up the entire system, and the other is issues related to smart contracts (**Figure 1**).

Threat analysis requires the selection of targets to work on, so here as a concrete example, we will delve into the issues to be addressed by using the example of a service that defines virtual currency transactions with a smart contract.

2.1 Issues of boundary design

In the case of Bitcoin, which uses a public blockchain, the validity of virtual currency transactions is ensured through prevention of double use with a method called Unspent Transaction Output (UTXO).

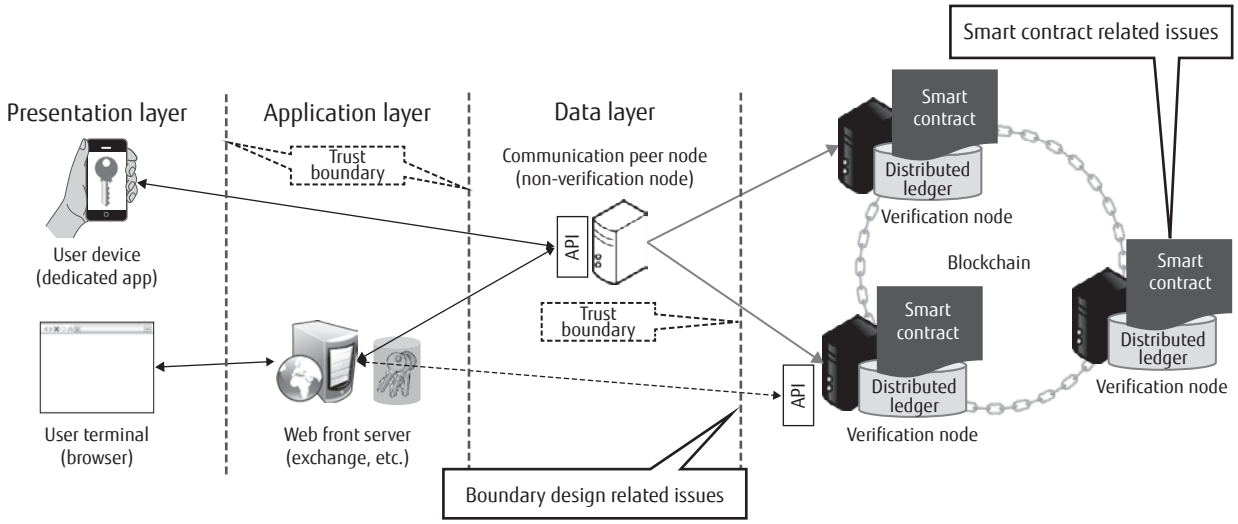
On the other hand, balance management by, among other means, Fabric and Ethereum smart contracts [note)], is not subject to interference by administrators, so the validity of transactions is ensured by the smart contract execution logic. However, if there are deficiencies on the source call information in the transaction even if the execution logic is correct, the validity of the transaction cannot be guaranteed, and mistakes become possible, such as the same transaction being executed multiple times. When there are different levels of reliability between the execution logic and the source call information, for example, a trust boundary is said to exist. Care must be taken in designing at each trust boundary from the viewpoint of ensuring validity. Yet, as awareness of the importance of doing so is still lacking, that design flaws do occur, giving rise to threats such as data tampering.

Blockchain systems that use smart contracts often implement a single function throughout the system. This results in vulnerabilities at trust boundaries. To ensure safety, it is necessary to analyze the threats of the entire system and take appropriate measures to prevent vulnerabilities from occurring at the trust boundaries. However, since the blockchain itself is a state-of-the-art technology, there are still few applied cases, and the establishment of a threat analysis method for systems that use a blockchain remains a task to be accomplished.

2.2 Issues of smart contract development

As mentioned in the previous subsection, Fabric

note) A generic term for a platform to run smart contracts.



API: Application programming interface

Figure 1
Overall configuration and issues of blockchain system.

trades assets through smart contracts. Thus, if there is a defect in the smart contract, this will result in problems such as transactions being conducted under incorrect conditions. For example, in The DAO Attack of 2016, despite the fact that the smart contract had been thoroughly reviewed, functions supposed to be called only once could be called multiple times, owing to improper use of functions and improper processing sequence of function calls. Exploitation of these defects resulted in the illegal transfer of virtual currency worth about 50 million dollars at that time.

Thus efforts to improve the quality of smart contracts are absolutely essential. Currently, the main approach is to determine whether problems might arise through reviews by blockchain and smart contract experts. Reviews by experts are highly effective and are a valid approach. However, reviews by experts being costly and requiring much labor, the possibility of things getting missed became a rising concern.

3. Threat analysis method for systems that use a blockchain

This section describes the threat analysis method for systems that use a blockchain, which was developed this time to solve the design issues described in subsection 2.1.

3.1 Threat analysis method

System-wide risk analysis was implemented in the five steps listed in **Table 1**, taking as reference the "Security Risk Assessment Guide for Industrial Control Systems" (hereafter, IPA Guide),[2] an analysis method for industrial control systems published by the Information-technology Promotion Agency, Japan (IPA). The IPA Guide is an analysis method aimed at dealing with high-priority risks.

The purpose of the risk analysis implemented this time is not analysis of the entire system, but threat extraction of the parts related to the blockchain. We thought it would be possible to fully ensure quality while achieving burden reduction by devising an effective verification method. Specifically, we changed the analysis methods, principally in step 5 "Risk analysis," and achieved burden reduction by analyzing only the parts related to the block chain in a top-down approach.

Generally, systems that use a block chain comprise multiple hierarchical structures. The blockchain corresponds to the lower layers, and by limiting the number of applications that use the blockchain in the upper layers, the number of functions used in the blockchain can be reduced. This obviates the necessity of analyzing all the functions of the blockchain exhaustively and allows the analysis target to be narrowed down. In top-down analysis, the same thing can occur in the process of case segmentation. By verifying whether these cases can be handled as the same case, duplication of cases can be minimized. Through the above, total analysis accuracy comparable to that of bottom-up analysis is achieved while reducing the analysis burden.

Moreover, in step 3, numerical expression of importance level is omitted, and in calculating the risk value based on the importance level in step 4, the risk value is determined by a simple method. As a result, compared with the IPA Guide, while the proposed approach requires detailed analysis of the blockchain, it limits the burden of risk assessment to the blockchain, allowing more energy to be focused on the extraction of blockchain-related threats.

Based on the IPA guide, we were able to perform threat analysis efficiently by concentrating on blockchain threat extraction. At the same time, the threats occurring on the blockchain were systematized, including some countermeasures. Through this systemization, we were able to confirm that blockchain

Table 1
Risk analysis steps performed based on the IPA guide.

Steps	Analysis Description	Details
1	Clarification of system configuration	Clarifies the target of risk analysis to allow understanding of the overall system structure.
2	Clarification of use cases	Clarifies the flow of data across the trust boundaries for each use case.
3	Determination of asset importance	Quantifies the risk of each asset.
4	Definition of business damage and its levels	Identifies the security requirements that need to be addressed.
5	Risk analysis	Performs risk analysis based on the security requirements and generates a list of items that require confirmation.

system developers can efficiently perform threat analysis specialized for blockchain.

3.2 Threat analysis results

By applying this threat analysis to an actual blockchain system, we were able to analyze threats efficiently. This subsection introduces some of the threats extracted as a result of analysis and how to prevent them.

1) Double execution of transfer

Fabric, which is the subject of this analysis, implements a mechanism that prevents the same transaction from being executed multiple times, so that the same request is not executed twice. Generally, when similar requests are repeatedly executed vis-a-vis a block chain, such as a retransmission attack, transactions with different identifiers occur. For example, if a user accidentally requests a transfer twice, separate transactions occur, so the transfer is made twice.

To prevent this problem, several methods may be considered. One is to introduce an interface that does not allow double transfers to the smart contract interface that performs the transfer processing. Another one is to check whether the application program has a function that prevents the user from requesting a transfer twice by mistake. Furthermore, the smart contract interface and processing content are checked against the processing content of the higher-level application to verify that the smart contract does not execute a double transfer.

2) Transfer from another person

The transfer process defined in the smart contract requires at least three elements: transfer source account, transfer destination account, and transfer amount. In the transfer process, it is necessary to prevent transfers from being requested by a party other than the owner of the transfer source account. To that end, a mechanism to check whether the requester is the owner of the transfer source account in the smart contract, and if the requester is not the owner, to prevent processing of the transfer, is required.

Depending on the system, there may also be the case where the owner of the transfer source account may want to delegate transfer execution authority to another person so that the other person can perform a transfer on behalf of the owner. In that case, it is necessary to prove that the request is from the owner of the transfer source account, for example by provision of the signature of the owner of the transfer source account. The signature must also be a one-time use signature that can be used only once in the smart contract, not a reusable signature. This can be achieved by using a time stamp or nonce (disposable random number) for the signature.

3) Issuance of virtual currency

The mechanism for consensus building between unspecified participants and suppressing fraudulent transactions is called mining, and virtual currency can be obtained as a reward for mining. In a blockchain where mining is performed, virtual currency is issued for each unit of mining work completed. On the other hand, a system where mining is not performed requires a mechanism for issuing the virtual currency handled in the system and making transfers to specific accounts. Usually, the issuance of virtual currency and its transfer to specific accounts is done by an administrator. If an administrator had malicious intent, he could illegally issue virtual currency and transfer it to his own account for personal gain. To prevent such human threats, technology such as multisig has been introduced so that a transaction cannot proceed unless approval by a least a set number of administrators is obtained.

Thus, even a transfer process involves processes with multiple vulnerabilities, which can be prevented with various methods. Comprehensively suppressing such vulnerabilities requires an awareness of the threat hidden in the entire system based on a grasp not only of analysis methods but also of the functions realized by smart contracts. Moreover, as threats vary depending on the system configuration and the functions realized by smart contracts, analysis is required each time a change is made.

4. Smart contract verification technology

To solve the issues listed in subsection 2.2 regarding smart contract development, we have developed smart contract verification technology based on static analysis for the three types of risks that are currently known. This section describes the developed verification technology and some of the risks detected using this technology.

4.1 Developed static analysis technology

Figure 2 shows the general flow of the static

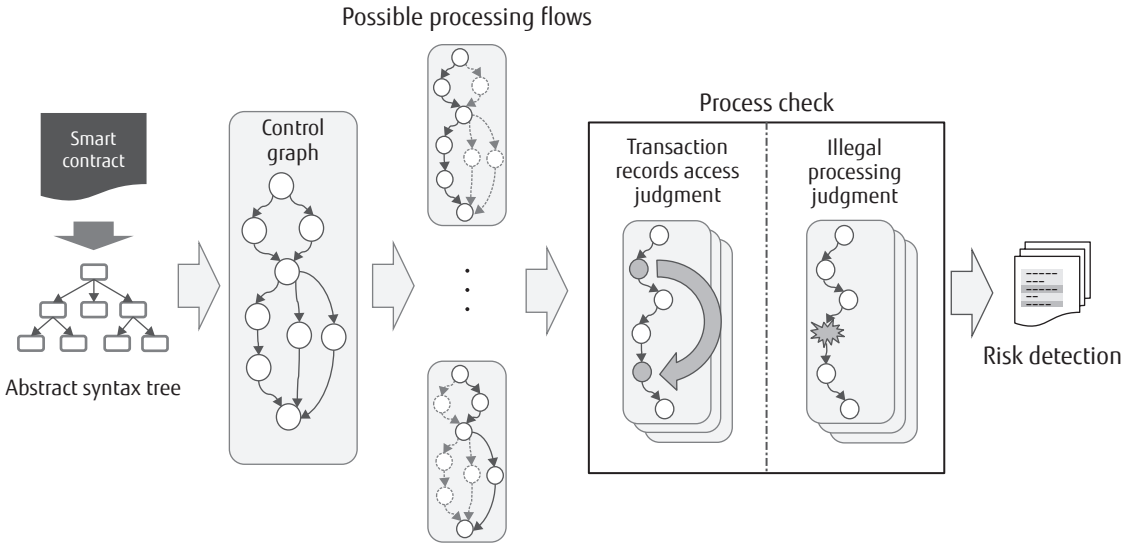


Figure 2
Flow of static analysis technology.

analysis technology. First, a control graph representing possible processing flows is generated from an abstract syntax tree in which the source code of the smart contract is mapped. Using this control graph, risks are detected by checking each process in turn. This technology identifies risks in each process by performing blockchain-specific transaction records access judgment and checks for any illegal processing according to predefined rules.

This static analysis technology can automatically detect risks such as those described in the next subsection without omissions. The result is reduction of developer review cost, shortening of the development process, and improvement of the quality of smart contracts. Improved quality of smart contracts will lead to the construction of highly secure blockchain systems that prevent erroneous transactions.

4.2 Examples of detected risks

This subsection describes the three types of risks that are the detection targets this time with specific examples.

1) Programming language dependent risks

In the case of Fabric, smart contracts are developed using general-purpose languages such as Go, Node.js, and Java. Using a general-purpose language has the advantage of reducing the learning costs for developers. On the other hand, languages designed specifically for the development of smart contracts such as Solidity[3] and Vyper[4] can limit functions that should not be used in smart contracts given the characteristics of blockchain. As there are no such restrictions in the case of general-purpose languages, there is the risk that developers using general-purpose languages will use such functions by mistake.

An example of functions that should not be used in smart contracts is random numbers. A smart contract is executed independently at each verification node and whether the result is the same for all the nodes is verified. However, if random numbers are included, the results are expected to vary depending on the environment. In that case, no consensus is reached among the verification nodes, and transactions are not performed correctly. To avoid such risk, languages designed for smart contracts such as Solidity do not use random numbers.

2) Fabric specifications dependent risks

There are Fabric specifications dependent risks.

For example, Fabric allows simultaneous read and write in a single transaction. However, it does not support read-your-writes consistency. Therefore, if the code description order is incorrect, data written in a transaction cannot be read in the same transaction. In that case, the data before new data is written is read, causing problems such as the transaction result being different from that intended by the user, or double

payment being executed.

3) Database specifications dependent risks

Fabric has a mechanism called the world state that stores the latest state of the blockchain. The world state is implemented using a database such as Apache CouchDB.[5] The smart contract references the world state when the latest information on the blockchain is required to execute a transaction. Thus, due to the influence of phantom reads,[6] which can occur under the specifications of Apache CouchDB, the latest transaction status may not be confirmed correctly and incorrect transaction may be executed in some cases. In this way, developers must pay attention to database specifications dependent risks.

Further, in addition to the risks specific to blockchain and blockchain platforms mentioned above, risks that pose problems in general programming also need to be considered during development. However, these being general issues, they are not covered by the technology described in this paper.

5. Conclusion

This paper described the threat analysis method and smart contract verification technology that we have developed. Regarding the threat analysis method, we confirmed that it makes it easy to analyze threats related to blockchain by changing the analysis approach for some analysis based on the IPA guide. We also confirmed that smart contract verification technology based on static analysis can automatically detect risks without omissions.

Based on these findings, going forward we will enhance the reliability of systems built by Fujitsu using the blockchain. Smart contract verification technology will first be applied to smart contracts for blockchain-related services provided by Fujitsu to contribute to the realization of highly secure services.

References

1) E. Androulaki et al.: Hyperledger Fabric: "A Distributed Operating System for Permissioned Blockchains." EuroSys, 2018.
https://arxiv.org/pdf/1801.10228.pdf
2) IPA: Security Risk Assessment Guide for Industrial Control Systems.
https://www.ipa.go.jp/files/000065768.pdf
3) Solidity.
https://solidity.readthedocs.io/en/v0.4.24/
4) Vyper.
https://vyper.readthedocs.io/en/latest/index.html
5) Apache CouchDB.
http://couchdb.apache.org/
6) H. Berenson et al.: A Critique of ANSI SQL Isolation Levels. SIGMOD Rec., 1995.
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-95-51.pdf

Fumihiko Kozakura
Fujitsu Laboratories Ltd.
Mr. Kozakura is currently engaged in research on blockchain technology development.

Shingo Fujimoto
Fujitsu Laboratories Ltd.
Mr. Fujimoto is currently engaged in research on blockchain technology development.

Yoshihide Nomura
Fujitsu Laboratories Ltd.
Mr. Nomura is currently engaged in research on development support technology for distributed environments and development support technology for blockchain systems.

Kazuhiro Yamashita
Fujitsu Laboratories Ltd.
Mr. Yamashita is currently engaged in research on development support technology for blockchain systems.