

Blockchain Security Analysis with Multi-Factor Authentication and Multi-Signature Mechanisms

Weifeng Li^a

School of Informatics, Xiamen University, Xiamen, China

Keywords: Blockchain Security, Multi-Factor Authentication, Multi-Signature Mechanisms, Smart Contract.


Abstract: Blockchain technology is renowned for its decentralization and transparency but faces significant security challenges, particularly in large-scale deployments where encryption and consensus mechanisms are vulnerable to attacks. This study aims to bolster blockchain security by employing advanced techniques, specifically focusing on multi-factor authentication (MFA) and multi-signature mechanisms. The research adopts a comprehensive approach that integrates Mythril's static code analysis, JUnit's dynamic testing, and Echidna's fuzz testing to identify and address vulnerabilities in smart contracts. Static code analysis is used to detect common vulnerabilities, dynamic testing ensures module functionality, and fuzz testing uncovers edge-case issues. The study demonstrates the effectiveness of MFA in mitigating risks associated with password leakage through static and one-time passwords, while the multi-signature mechanism enhances security by requiring multiple approvals for transactions. Experimental results on a publicly available smart contract dataset reveal that these security enhancements substantially reduce security incidents and improve system stability. These findings offer practical solutions for optimizing blockchain security and provide a solid foundation for future research on safeguarding blockchain applications in complex scenarios.

1 INTRODUCTION

Blockchain technology, with its decentralization, immutability, and transparency, has shown great potential in finance, healthcare, and other fields. However, security and privacy issues in large-scale applications are gradually emerging. Vulnerabilities in cryptographic algorithms and consensus mechanisms on which blockchain relies may lead to security threats such as double payments and transaction tampering. Therefore, systematically analyzing the security issues of blockchain technology, identifying potential threats, and exploring effective countermeasure strategies are of great significance in safeguarding the security of blockchain systems and promoting their wide application (Zhang et.al, 2019).

In recent years, extensive research on blockchain security has been conducted in academia and industry, focusing on improving the system's resistance to attacks through cryptographic techniques, consensus algorithms, and security protocols. Many studies have focused on improving

consensus mechanisms, enhancing the security of smart contracts, and cryptographic security. For example, the evolution of consensus algorithms such as Proof of Work (PoW) and Proof of Stake (PoS) (Ferdous et.al, 2021), as well as the introduction of technologies such as non-interactive zero-knowledge proofs and hash-chain storage, have significantly improved the security and scalability of blockchains. Joseph Bonneau et al. provided the first systematic exposition of Bitcoin and other cryptocurrencies, analyzed the anonymity problem, and reviewed the privacy enhancement methods (Bonneau et.al, 2015). Ghassan Karame systematically outlined and analyzed the security provisioning of the blockchain in Bitcoin, including risks and attacks in Bitcoin-like digital currency systems (Karam, 2016). They also described and evaluated mitigation strategies to eliminate some of the risks. Mauro Conti et al reviewed the security and privacy of Bitcoin, including existing vulnerabilities that lead to various security risks during the implementation of the Bitcoin system (Conti et.al, 2018). Li et al. investigated the security risks of popular blockchain

^a <https://orcid.org/0009-0002-2306-4130>

systems, reviewed cases of attacks on blockchains, and analyzed the vulnerabilities exploited in these cases (Li et.al, 2020). Dasgupta et al provide a detailed categorization of blockchain security issues, covering everything from vulnerabilities in cryptographic operations to possible threats posed by quantum computing (Dasgupta et.al, 2019). In addition, some studies propose specific ways to counter these threats, such as using more secure elliptic curve algorithms to defend against cryptographic attacks (Zhang et.al, 2022). Other studies focus on security issues at the blockchain network level, such as encryption of inter-node communication and measures to prevent Distributed Denial of Service (DDoS) attacks (Sousa and Monteiro, 2018). As the application scenarios of blockchain technology continue to expand, especially in the fields of finance and the Internet of Things (IoT), the requirements for its security are getting higher and higher. Therefore, in-depth understanding and solving the security problems of blockchain technology in different application scenarios has become an important direction of current research. Therefore, addressing blockchain security in various application scenarios has become a key research direction.

The primary objective of this research is to systematically analyze the security challenges associated with blockchain technology and propose effective methods to address these issues. This study provides a comprehensive overview of security threats, classified based on existing literature, and emphasizes a comparative analysis of various cryptographic algorithms and consensus mechanisms. It further explores the security assessment of smart contracts through methods such as static code analysis, dynamic testing, and fuzz testing. Additionally, the research highlights the implementation of multi-signature and multi-factor authentication (MFA) to bolster the security of critical operations. The findings of this study reveal that the choice of cryptographic algorithms, the robustness of consensus mechanisms, and network-level protection measures are crucial for enhancing overall blockchain security. By focusing on these areas, the study offers substantial security improvements for practical blockchain applications and serves as a valuable reference for future research in blockchain security, particularly in complex application scenarios.

2 METHODOLOGIES

2.1 Dataset Description and Preprocessing

The dataset used in this study is mainly derived from publicly available smart contract platforms, such as Etherscan and smart contract repositories on GitHub (Etherscan, 2015). The dataset contains multiple types of smart contracts, such as token contracts, decentralized financial protocols, and governance system contracts. To ensure smooth experiments, the data preprocessing is done by removing extraneous data, standardizing the format of the contract code, and ensuring its compatibility with security testing tools such as Mythril and Echidna. In addition, duplicate or deprecated contracts in the dataset are filtered out to improve the accuracy of the experiments.

2.2 The Proposed Methodology

The objective of this research is to enhance the security and stability of blockchain smart contracts through advanced technological tools. This study integrates static code analysis, dynamic testing, and fuzz testing, while also incorporating multi-signature mechanisms and multi-factor authentication to bolster security. The research methodology, outlined in Figure 1, involves several key stages: identifying security vulnerabilities, conducting code testing and verification, and implementing security enhancements. The process begins with static code analysis to detect potential vulnerabilities. This is followed by dynamic testing to assess contract behavior under various conditions. Next, fuzz testing is employed to verify the robustness of the contracts by exposing them to a range of inputs. Finally, the integration of multi-signature mechanisms and multi-factor authentication is applied to further secure the system.

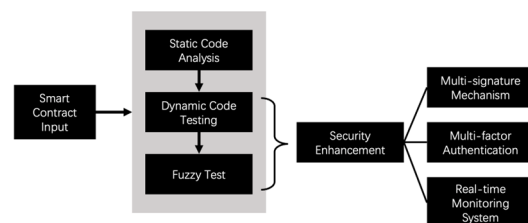


Figure 1: Flowchart of the study for this model (Picture credit: Original).

2.2.1 Static Code Analysis

Mythril is a powerful static analysis tool specifically designed to analyze the security of smart contract codes. Its main features include code syntax checking, pattern matching, and symbolic execution to effectively identify common vulnerabilities in contracts, such as reentry attacks and integer overflows. This study uses Mythril to perform an in-depth security analysis of the contract, generate a detailed vulnerability report, and fix the vulnerabilities according to the recommendations in the report. The advantage of Mythril lies in its ability to identify potential vulnerabilities before the execution of the contract, avoiding losses caused by exposing the problems after the code goes live. During the implementation process, this study first inputs the contract into Mythril for static analysis, and the generated vulnerability report provides a reference for subsequent dynamic testing.

2.2.2 Dynamic Testing

Dynamic testing is critical to ensure that the code behaves properly at runtime. This study adopted the JUnit framework to conduct unit tests for different modules of the smart contract. The advantage of JUnit is that it can automatically run pre-written test cases every time the code is changed so that new vulnerabilities in the code can be captured promptly. This paper wrote detailed test cases for each contract module to ensure that each module works properly in an independent environment and effectively reduces the testing bias caused by external dependencies. During the testing process, JUnit can help researcher verify the boundary conditions and exception handling of the contract to ensure its stability in real applications.

2.2.3 Fuzz Testing

Fuzz testing is a random input testing method for verifying the robustness of smart contracts. This study uses the Echidna tool to generate a large number of random inputs to test whether a smart contract can satisfy predefined security properties. The advantage of Echidna is that it can generate a wide range of different random test cases, which helps researcher to discover potential vulnerabilities that cannot be covered by regular tests. This study uses Echidna to perform fuzzing tests on several smart contracts, and by analyzing the generated test reports, this study has promptly identified and fixed several boundary case vulnerabilities. The failure

cases provided by Echidna also provide valuable references for further optimizing the contract design.

2.2.4 Security Enhancement

To further enhance the security of the system, this study introduces multiple signature mechanisms and MFA in the smart contract. The multi-signature mechanism requires multiple authorized signers to participate in the transaction or operation together, and the transaction can only be executed after the set signature threshold is reached. This study sets up a signer manager class in the system to collect and verify the public and private keys of the signers, thus ensuring the security of the transaction. Multi-factor authentication, on the other hand, provides users with double protection by combining static passwords and one-time passwords (TOTP). Especially when performing sensitive operations, MFA effectively prevents the security risks associated with a single password leakage.

2.3 Implementation Details

In the implementation of the system, this study used Java environment for development. To improve the accuracy of the model, this study performed data enhancement operations and used the TOTP algorithm to generate one-time passwords. In the experiments, the following hyper-parameter settings were used: the depth of static analysis was 3 layers, the number of inputs for fuzz testing was 1000 random samples, and the timeout for MFA was 60 seconds.

3 RESULT AND DISCUSSION

This chapter will summaries and analyze the experimental results, mainly discussing the results of static code analysis, dynamic testing, and fuzz testing, as well as analyzing the system security enhancement after the introduction of the multi-signature mechanism and multi-factor authentication.

3.1 Static Code Analysis Results

As shown in Figure 2, the static code analysis of the smart contract was conducted using Mythril, and the results show that the original smart contract has several security vulnerabilities, mainly including reentry attacks and integer overflow problems. These vulnerabilities were detected before the contract went live, thus avoiding potential financial losses. Figure 2

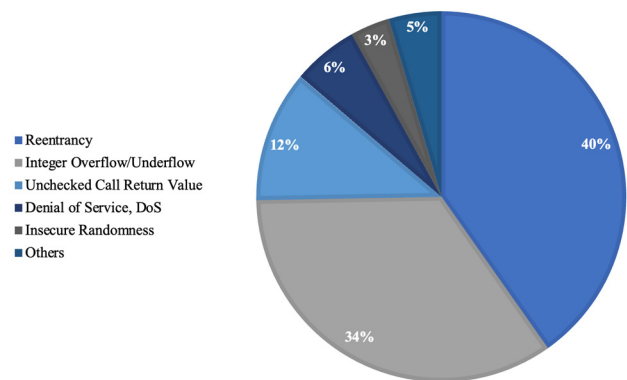


Figure 2: Distribution of different types of vulnerabilities in static code analysis (Picture credit: Original).

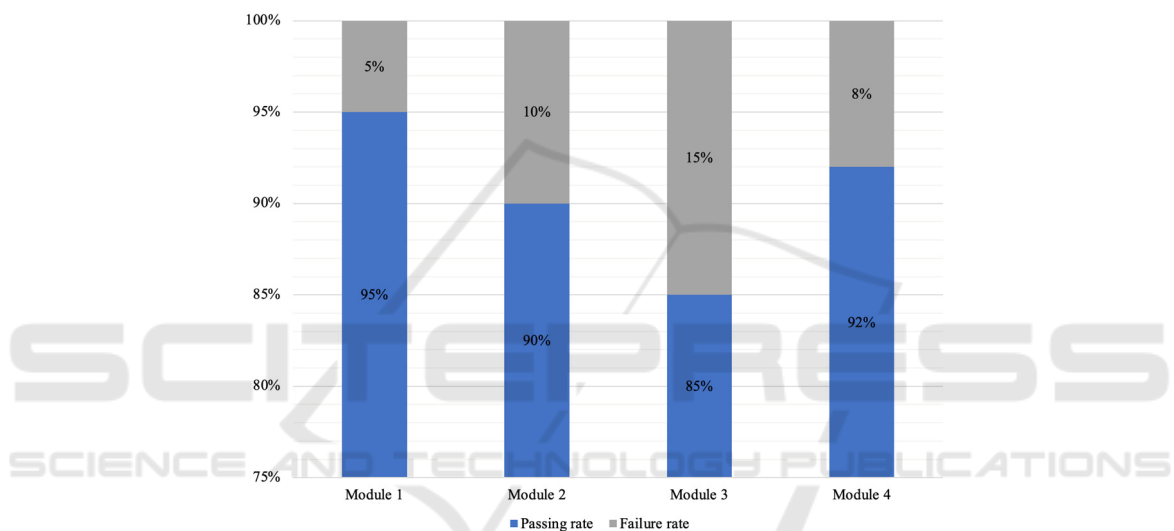


Figure 3: JUnit dynamic test results show (Picture credit: Original).

shows the detection results of different types of vulnerabilities, with reentry attacks accounting for 40%, integer overflow for 34%, and the rest being other types of vulnerabilities. With these vulnerability reports, the code was further modified to significantly reduce the potential attack surface.

3.2 Dynamic Test Results

Figure 3 shows the results of dynamic testing using JUnit after fixing the vulnerabilities found in the static analysis. The test covers all the modules of the contract and simulates a variety of operating environments to ensure the stability and correctness of each module when it is executed independently. The tests show that the repaired smart contract performs well under all boundary conditions and exception-handling situations, and no new

vulnerabilities were found. This result shows that JUnit automated testing can detect potential problems introduced by code changes promptly.

3.3 Fuzz Test Results

Fuzz testing was conducted using Echidna, by generating a large number of random inputs, Echidna detected 3 boundary case security vulnerabilities out of 1000 test samples. These vulnerabilities, although difficult to catch in normal testing, were exposed and fixed by fuzz testing. This test result verifies the effectiveness of fuzz testing in identifying potential boundary vulnerabilities and further improves the security of smart contracts.

Table 1: Comparison of security before and after introducing multi-signature and multi-factor authentication.

Metric	Before Introduction	After Introduction
Number of Security Incidents	20 incidents	5 incidents
Authentication Strength	Medium	High
Transaction Risk	High	Low
Single Password Attack Risk	High	Very Low
Overall System Stability	Fair	Significantly Improved

3.4 Effectiveness of Security Enhancement Measures

After adding multi-signature and multi-factor authentication, the overall security of the system is significantly improved. Table 1 shows the security comparison before and after the introduction of these mechanisms. After adding multi-signature, important operations such as fund transfers must be co-signed by multiple authorized persons, effectively preventing the risks caused by the failure of a single signature. Meanwhile, multi-factor authentication prevents account risks caused by password leakage through the combination of static and one-time passwords. Overall, this chapter comprehensively improves the security of the smart contract system through static analysis, dynamic testing, fuzz testing, and the introduction of security mechanisms to ensure the robustness and security of the system in various scenarios.

4 CONCLUSIONS

The primary objective of this study is to analyze and enhance the security of blockchain smart contracts. The proposed methodology integrates static code analysis, dynamic testing, and fuzz testing, while also introducing multi-signature mechanisms and multi-factor authentication to strengthen system security. The effectiveness of this approach was evaluated through extensive experiments. The results demonstrate that static code analysis effectively identifies common security vulnerabilities, dynamic testing ensures code stability during execution, and fuzz testing uncovers potential vulnerabilities at the boundaries. Additionally, the incorporation of security enhancements significantly mitigates the risks associated with high-risk operations. Future research will focus on addressing the security challenges of blockchain systems in diverse application scenarios, particularly in IoT and financial sectors, to further refine and improve smart contract protection mechanisms. Additionally, exploring the potential impact of quantum computing

on blockchain cryptographic algorithms will be a crucial area of investigation.

REFERENCES

- Bonneau, J., Miller, A., Clark, J., et al. 2015. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. *IEEE symposium on security and privacy*, 104-121.
- Conti, M., Kumar, E.S., Lal, C., et al. 2018. A survey on security and privacy issues of bitcoin. *IEEE communications surveys & tutorials*, 20(4), 3416-3452.
- Dasgupta, D., Shrein, J.M., & Gupta, K.D., 2019. A survey of blockchain from a security perspective. *Journal of Banking and Financial Technology*, 3(1), 1-17.
- Etherscan, 2015. Etherscan API Documentation. Retrieved from: <https://docs.etherscan.io/>.
- Ferdous, M.S., Chowdhury, M.J.M., Hoque, M.A.A., 2021. survey of consensus algorithms in public blockchain systems for crypto-currencies. *Journal of Network and Computer Applications*, 182, 103035.
- Karame, G., 2016. On the security and scalability of bitcoin's blockchain. *Proceedings of the ACM SIGSAC conference on computer and communications security*, 1861-1862.
- Li, X., Jiang, P., Chen, T., et al. 2020. A survey on the security of blockchain systems. *Future generation computer systems*, 107, 841-853.
- Sousa, A.D., & Monteiro, E.B., 2018. Blockchain from the analysis of cases of use in the corporate environment: A systematic review. *Journal of Information Systems and Technology Management*, 15(2), 1-18.
- Zhang, K., Manzoor, A., Chang, V., Rodrigues, J.J.P.C., & Mazurczyk, W., 2022. A survey on blockchain technology and its applications: Research issues and challenges. *Blockchain: Research and Applications*, 3(2), 100012.
- Zhang, R., Xue, R., Liu, L., 2019. Security and privacy on blockchain. *ACM Computing Surveys*, 52(3), 1-34.