Journal of Electrical Engineering and Automation (ISSN: 2582-3051)
www.irojournals.com/iroeea/

# Design of an Ethereum Security Analysis for Unpredictable State System - An Overview

**C. Anand**

Department of CSE, K.S.R. College of Engineering, Tiruchengode, India

**E-mail:** anandsione@gmail.com

## Abstract

As a powerful tool for building framework and autonomous system across various domains, smart contracts are used to maintain security analysis in a better way. However, owing to the decentralised structure of the blockchain on which they are built, a number of concerns have surfaced about weaknesses in their programming, that according to their unique characteristics, might have (and have already had) a very large economic effect. This essay surveys the whole scope of smart contract security issues and the cutting edge of freely accessible security software.

**Keywords:** Smart contracts, blockchain, ethereum, security development, virtual machine

## 1. Introduction

The major cryptocurrencies in the blockchain technology sector have recently come into focus due to the field's rapid expansion. Ethereum stands out because its Virtual Machine (EVM) enables smart contracts, i.e., distributed programmes that regulate the flow of Ether digital money. This Turing-complete application, make it possible to describe a wide variety of monetary uses. However, this level of expressiveness comes at the expense of substantial semantic complexity, which in turn raises the likelihood of programming mistakes. Additionally, recent assaults using flaws in intelligent contract implementations need the development of formal verification approaches for smart contracts. Abstraction methods tailored to the EVM's semantics, a detailed description of the security attributes wished to be seen, and rock-solid semantic groundwork to do this are required [1-5].

Even in the absence of trustworthy third parties, blockchain technologies provide the promise of safe distributed computing. A distributed ledger is the backbone of this system,


*DOI: https://doi.org/10.36548/jeea.2022.3.007*
*Received: 22.08.2022, received in revised form: 19.09.2022, accepted: 04.10.2022, published: 14.10.2022*

recording all transactions and the current status of each account and being protected by a mix of incentives and encryption. Using the programming language that is native to the underlying cryptocurrency, programmers may create sophisticated distributed calculations that are grounded in transactions [6-9].

### 1.1 Ethereum

Ethereum is a decentralised blockchain-based digital money platform. Similar to Bitcoin's proof-of-work consensus process, network members broadcast transferring node to blockchain. To determine the system's state, or the global state, the individual accounts that make up the system as a whole are studied. It stores information about the account's current balance, whereas a contract account stores both the contract's code and permanent storage. The account balances are shown in the virtual currency's smallest denomination [10].

Changes to the status of the system may occur when either new contract accounts are created, or an existing account is called. Storage for the performance may be modified or additional transactions may be performed during the contract's execution; this context refers to internal transactions [11 - 13].

### 1.2 Blockchain

Blockchain refers to a distributed ledger that may be used to track transactions and verify their authenticity in a way that is safe, permanent, easy to use, and verifiable [14]. In addition, cryptocurrencies like Bitcoin and Ethereum rely on this technology. Despite its roots in financial transactions, blockchain technology has the potential to impact other sectors, like 5G and networking [15]. In the same way that hardware drivers make it possible for programmes to work together with hardware, smart contracts may make it possible for IoT apps to connect with one another [16]. Additionally, 5G networks using decentralised ledgers may benefit from the enhanced security that can be provided by smart contracts. Ethereum's programmability stems from the platform's support for the creation and execution of smart contracts. Figure 1 shows simple blocks of smart contract. In addition, recent high-profile cases have hinted that some blockchain intelligent contracts may contain security flaws in their programming that might be exploited to steal money [17]. Given that smart contracts are meant to be "immutable", this poses a significant challenge. The inability to modify deployed contract code prevents exploits from being fixed.

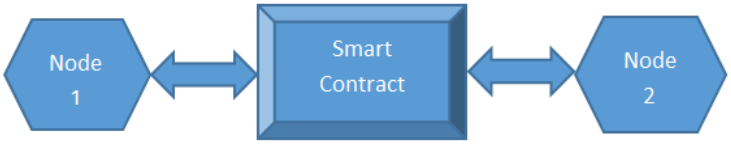


**Figure 1.** Simple blocks of Smart Contract

### 1.3 Motivation

Although the notion of smart contracts is not new, it has lately seen a surge in interest. This concept of "intelligent contracts" has been around since 2007, but it has really taken off with the advent of Bitcoin and the decentralised blockchain in 2009. Decentralized computer programmes known as "smart contracts" may update the system status recorded in a distributed ledger called a "blockchain" if everything goes according to plan. Smart contracts can be used in a wide range of contexts, from identity management and electronic voting through Internet of Things (IoT), online gaming, and medical records. Although it's most often linked with Ethereum, several other systems now employ it as well, including Hyperledger and Corda. While Ethereum is the platform of choice for this endeavour, many of the proposed security measures are applicable to other systems as well.

## 2. Background

Transactions in smart contracts are managed entirely by computer code. Recently, the input to the relevant smart contract is provided by users through payload data included in transactions. To be more precise, a contract is defined as a set of user-invokable operations. It is also possible for one contract to initiate the execution of another contract through a "call" command. Such an important command communicates data in a manner through a sender that it must submit a fee, which will be derived from the computational cost of the contract, in order for the contract to be executed.

Additional contracts on the Ethereum blockchain may also contact the arrangements [18]. Even though reports may react to incoming transactions, they cannot initiate any new trades of their own. Smart contracts are attractive to cybercriminals because they may be used to control and store Ether.

### 2.1 Concept of a Distributed Ledger (DL)

A distributed ledger is a kind of data structure that is duplicated and synced across various nodes within a network, along with a set of techniques for working with those copies.

Immutability, resistance to censorship, distributed administration, and the lack of need for a trusted communication networks are some of the most striking features of a Distributed Ledger Technology (DLT). Many different types of DLT exist, each based on a different data structure for various security process.

## 3. Security for Unpredicted System

### 3.1 Hypercritical Operation

It is required to know that most attacks are associated with value transformation activities, and that these operations include many EVM instructions. Some of the most often exploited database commands are establishing transactions, terminating transactions, and injecting code. Both the overall picture and the finer points of some of the most important learning are shown. There are two types of addresses on the blockchain: user addresses and contract addresses. Users transmit needs from their user address to the contract address in order to execute a smart contract. Given the immutability of intellectual agreements and their use in highly sensitive fields like banking and medicine, this poses a serious security concern [19].

### 3.2 Unpredictable condition

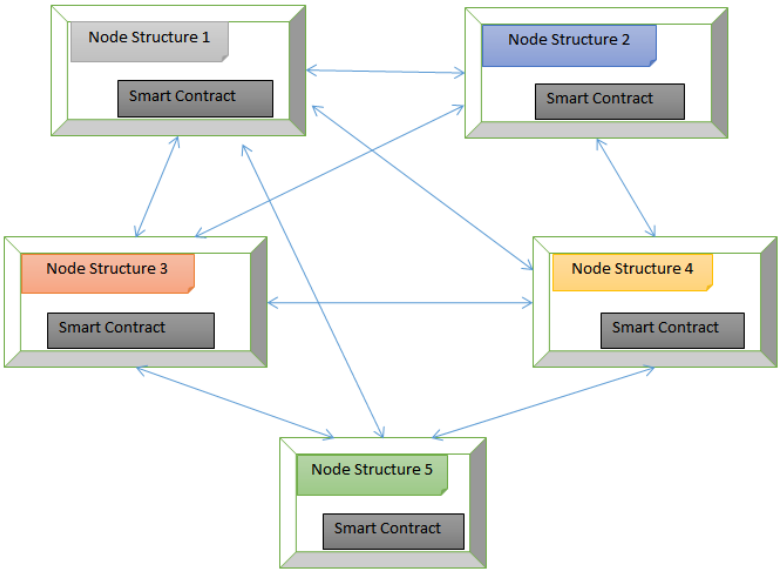


**Figure 2.** Fully connected decentralized system

The fields and Ether balance sheet of a smart contract define its state. Thus, the transaction will be processed quickly which is not guaranteed. Further complicating matters is the fact that two miners might mine the same valid block at the same moment, leading to a chain fork. Thus, some miners will attempt to selectively add their blocks to one of the two

strings while ignoring the other. E.g., agreements with no set end date. Contracts that make use of dynamic libraries are another example. It would be possible for a malevolent party to alter such arrangements and fool the victim into thinking everything was OK when they called in [20, 21]. This fully connected decentralized system structure has been shown in figure 2.

### 3.3 Examining the dynamics

The analysis is performed in real time, while the software is being used. It identifies flaws that traditional static analysis would have missed due to factors like obtuse coding or secure packaging. In addition, it validates the security flaws identified by static analysis.

#### 3.3.1 Analyzing symbols

By assigning many values to monitor their subsequent approach, symbolic execution of the contract may be performed based on various execution traces.

#### 3.3.2 Verification of False and True Positives

This outcome of the in-depth research is double-checked using real-world information.

### 3.4 Threats to Smart Contracts that can have a Major Impact

In this part, two high-profile assaults on smart contracts' security that had major financial repercussions are studied. Because of the high value of the funds (in the form of cryptocurrency) involved, smart contracts are often the target of hackers. If a smart contract has a flaw that may be exploited by hackers, that flaw cannot be fixed because of the blockchain's permanent nature. Any intelligent warranty that relies on the smart contract will be voided if the contract is "destroyed" (marked as useless), as this section demonstrates. Therefore, it is crucial to conduct tests on the safe development technique before integrating them into the blockchain [22].

### 3.5 Setup of a Node on the Ethereum Network

An Ethereum smart contract's original source code may be obtained in one of the two methods. Even though using Application Programming Interface (API) is more user-friendly, it is decided to go with the alternative approach to building ESAF due to its flexibility and scalability [23].

### 3.6 Extraction from smart contracts

It's important to describe the process of the following.

**To wit: Trie, state case**

The current state of the whole blockchain can be found in this trie, making it the most important Ethereum. This dynamic database stores a unique Ethereum account and is updated in real time.

**Archival triad:**

There is a separate trial for each Ethereum account, in which all of that account's data may be stored.

### 3.7 Analysis of Smart Contract Weaknesses

The contracts using the meta-tool are examined at this stage. It demonstrates the steps used to be ready for the analysis up to the point when iterations on the chosen contracts are being performed. The ranges that will be studied may be seen.

### 3.8 Brief evaluation of performance

Some preliminary performance tests of the framework are included in this section. The testing will use the one hundredth contract to be created on the Ethereum network. To that end, ESAF has already retrieved such a contract from the blockchain and archived it. Parallelism options were kept out of the device to make it simpler to employ in the latest investigation [24].

## 4. Challenges and Future Work

- Responsible disclosure is typically not possible due to the immutability of the contract code upon deployment and the anonymity of contract owners.
- Accordingly, it looks difficult to deal with susceptible contracts method to correct the problems discovered in previous existing research work [25, 26]. In this case, the only option left to the contract owners is to determine the vulnerability through transfer from one node to another contract by various predicted domain sector.
- This is challenging since the exposed contract's address may be utilised in other contexts.

- As expected, vulnerabilities are detected in the majority of contracts during the scanning. This is because earlier compiler versions are included in the downloaded portion of the intelligent contract database. There are limits to how well network-wide scanning can function, such as, it is difficult to download a big number of contracts, and the high price of bringing nodes into sync with the primary factors.

## 5. Conclusion

This article provides Ethereum Security, a platform for performing automated security audits and creating exploits for the Ethereum smart contract environment. This method considers and successfully tackles the complexity involved in assessing commercially relevant smart contracts in the real world. The unique contribution of this article is the use of an effective analysis method that detects previously unknown vulnerabilities and assaults and produces trustworthy exploits for the discovered flaws. It concentrates on studying how to keep the smart contracts safe in cutting-edge work. It is believed that the wide array of development tools available today might be overwhelming for newcomers and make it hard to know where to begin using them. Therefore, this kind of text might be used as a springboard. It seems clear that two of the most economically damaging attacks against smart contracts are also reasonably easy to execute technically.

## References

[1] L. Yu, W.-T. Tsai, G. Li, Y. Yao, C. Hu, and E. Deng, “Smart-contract execution with concurrent block building,” in 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE). IEEE, 2017, pp. 160–167.

[2] Z. Gao, L. Xu, L. Chen, N. Shah, Y. Lu, and W. Shi, “Scalable blockchain based smart contract execution,” in 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2017, pp. 352–359.

[3] T. Min and W. Cai, “A security case study for blockchain games,” arXiv preprint arXiv:1906.05538, 2019.

[4] I. Nikolic, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, “Finding the greedy, prodigal, and suicidal contracts at scale,” arXiv preprint arXiv:1802.06038, 2018.

[5] P. Zhang, D. C. Schmidt, J. White, and G. Lenz, “Blockchain technology use cases in healthcare,” in Advances in Computers. Elsevier, 2018, vol. 111, pp. 1–41.

[6] K. N. Griggs, O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," Journal of medical systems, vol. 42, no. 7, p. 130, 2018.

[7] F. Knirsch, A. Unterweger, G. Eibl, and D. Engel, "Privacy-preserving smart grid tariff decisions with blockchain-based smart contracts," in Sustainable Cloud and Energy Services. Springer, 2018, pp. 85–116.

[8] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: towards sustainable local energy markets," Computer Science-Research and Development, vol. 33, no. 1-2, pp. 207–214, 2018.

[9] C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, and M. Bertoncini, "Blockchain based decentralized management of demand response programs in smart energy grids," Sensors, vol. 18, no. 1, p. 162, 2018.

[10] M. Mylrea and S. N. G. Gourisetti, "Blockchain for smart grid resilience: Exchanging distributed energy at speed, scale and security," in 2017 Resilience Week (RWS). IEEE, 2017, pp. 18–23.

[11] Tovanich, N., Heulot, N., Fekete, J.-D., and Isenberg, P. (2021). "Visualization of Blockchain Data: A Systematic Review," in IEEE Transactions on Visualization and Computer Graphics, 27, 3135–3152. doi:10.1109/ TVCG.2019.2963018

[12] [Dataset] Trail of Bits (2020). (Not So) Smart Contracts. Available at: https:// github.com/crytic/not-so-smart-contracts (Accessed August 7, 2021).

[13] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba,"Blockchain technology innovations," in Technology and Engineering Management Conference (TEMSCON), 2017 IEEE. IEEE, 2017, Conference Proceedings, pp. 137–141.

[14] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in Software Architecture (ICSA), 2017 IEEE International Conference on. IEEE, 2017, pp. 243–252.

[15] G. W. Peters and E. Panayi, "Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money," in Banking beyond banks and money. Springer, 2016, pp. 239–278.

[16] J. Gao, H. Liu, C. Liu, Q. Li, Z. Guan, and Z. Chen, "Easyflflow: Keep ethereum away from overflflow," in Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings. IEEE Press, 2019, pp. 23–26.

[17] I. Sergey, A. Kumar, and A. Hobor, “Scilla: a smart contract intermediate-level language,” arXiv preprint arXiv:1801.00687, 2018.

[18] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, “Reguard: fifinding reentrancy bugs in smart contracts,” in Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings. ACM, 2018, pp. 65–68.

[19] L. W. Cong and Z. He, “Blockchain disruption and smart contracts,” The Review of Financial Studies, vol. 32, no. 5, pp. 1754–1797, 2019.

[20] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and Y. Smaragdakis, “Madmax: Surviving out-of-gas conditions in ethereum smart contracts,” Proceedings of the ACM on Programming Languages, vol. 2, no. OOPSLA, p. 116, 2018.

[21] Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. Future Gener. Comput. Syst. 2020, 105, 475–491.

[22] Dingman, W.; Cohen, A.; Ferrara, N.; Lynch, A.; Jasinski, P.; Black, P.E.; Deng, L. Defects and vulnerabilities in smart contracts, a classifification using the NIST bugs framework. Int. J. Netw. Distrib. Comput. 2019, 7, 121–132.

[23] Grishchenko, I.; Maffei, M.; Schneidewind, C. A Semantic Framework for the Security Analysis of Ethereum Smart Contracts. In Principles of Security and Trust; Bauer, L., Küsters, R., Eds.; Springer: Cham, Switzerland, 2018; pp. 243–269.

[24] Praitheeshan, P.; Pan, L.; Yu, J.; Liu, J.; Doss, R. Security analysis methods on Ethereum smart contract vulnerabilities: A survey. arXiv 2019, arXiv:1908.08605.

[25] Huang, Y.; Bian, Y.; Li, R.; Zhao, J.L.; Shi, P. Smart contract security: A software lifecycle perspective. IEEE Access 2019, 7, 150184–150202.

[26] He, D.; Deng, Z.; Zhang, Y.; Chan, S.; Cheng, Y.; Guizani, N. Smart Contract Vulnerability Analysis and Security Audit. IEEE Netw. 2020, 34, 276–282.

## Author’s biography

**C. Anand** has completed B.E. in EEE and M.E. in CSE. He has more than 12 years of teaching experience and has published more than five international journals on the field of wireless sensor networks. He has guided about 10 PG students and currently he is working as an Assistant Professor in the department of computer science and Engineering at K.S.R. College of Engineering, Triuchengode, Tamil Nadu, India. He is a lifetime member of ISTE.