A Formal Framework for security assessment of Ethereum Smart Contracts[*]

Chiara Braghin[1][0000−0002−9756−4675], Elvinia Riccobene[1][0000−0002−1400−1026], and Simone Valentini[1][0009−0005−5956−3945]

Computer Science Department,
Università degli Studi di Milano, via Celoria 18, Milan, Italy
{chiara.braghin, elvinia.riccobene, simone.valentini}@unimi.it

Context and Motivations. Blockchain has shown to be a versatile technology with applications ranging from financial services and supply chain management to healthcare, identity verification, and beyond. Thanks to the usage of smart contracts, blockchain can streamline and automate complex processes, eliminating the need for intermediaries and reducing administrative overhead.

As blockchain-based smart contracts gain mainstream adoption, the demand for reliable and secure smart contract design and development becomes increasingly vital. Smart contracts are programs that govern high-value financial assets, but they carry a substantial risk due to their public availability, immutability, and the ability for anyone to execute them. For example, the infamous DAO hack drained $70 million worth of Ether from a vulnerable smart contract that was not properly verified [2]. Costly vulnerabilities and exploits can seriously hinder trust and acceptance in the blockchain ecosystem. Smart contracts often handle valuable assets and execute critical functions, making them attractive targets for attackers. Design-time error detection would allow developers to catch and rectify issues before the smart contract is deployed on the blockchain, enhancing the overall security, reliability, and trustworthiness of smart contracts.

Formal verification may contribute to the overall maturity and spread of blockchain technology by providing a robust methodology for security assessment of smart contracts. The field of formal verification for smart contracts has made notable progress but still faces challenges.

State of the Art. Several approaches and tools have been developed to formally verify smart contracts, particularly those written in languages like Solidity. However, existing tools have several limitations: bytecode-based tools cannot reason about contracts at design time, some tools use complex notations requiring a strong mathematical base that discourages many designers or engineers, and others target only a limited number of vulnerabilities [8, 9]. Therefore, challenges and research areas still require further development and improvement. For example, the languages and tools used for formal verification may not fully capture the specific features and behaviors of smart contracts. In addition, while formal verification tools can prove simple properties and catch common vulnerabilities, proving complex properties or verifying more sophisticated aspects of a smart contract's behavior remains challenging. Scalability is also a relevant issue: formal verification can be computationally expensive and time-consuming, especially for complex smart contracts or large codebases. In addition, smart contracts often interact with external systems, such as oracles and other smart contracts. Formal verification tools often struggle to model and verify the behavior of such external dependencies.

Among the most relevant approaches, Certora [1] is a formal verification tool specifically designed for Ethereum smart contracts. It supports the verification of correctness properties and functional specifications. Other relevant works are remarkable for their usage of symbolic execution, like Oyente [7], which performs a symbolic execution analysis on Ethereum smart contracts. Tools like VerX [10] and Securify [11] leverage abstract interpretation and dependency graphs to analyze smart contracts and verify the correctness of different properties expressed in temporal logic.

Research contribution. Our work aims to explore the potential of using the Abstract State Machine (ASM) formal method [4, 5] and its supporting toolset ASMETA [1] for the specification and

[*] This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU .

[1] https://asmeta.github.io/

verification of Ethereum smart contracts written in Solidity. Our long-term vision is to build a practical verification framework for security assessment of smart contracts.

To this aim, in [6] we formalized the Ethereum Virtual Machine (EVM) and key language primitives to enable functional correctness proofs. We developed libraries within the ASMETA framework that allows specifying Solidity contracts in ASMs, simulating their behavior, and formally verifying key properties. As demonstrated through examples [2], this enables detecting at design-time various vulnerabilities, such as access control issues, or reentrancy attacks. The ASMETA toolset allows different forms of model analysis. In particular, model verification is possible by verifying properties expressed in temporal logic: the model checking `AsmetaSMV` maps ASM models to the model checker `NuSMV`: the tool will check if the property holds during all possible model executions.

Given the heterogeneous nature of the possible types of vulnerabilities (e.g., unchecked external codes, gas limit and out-of gas issues, timestamp dependencies reentrancy attacks, unintended exposure of sensitive data, or errors in the flow logic), we are currently working to verify *intra-contract* properties, i.e., logical errors inside the given smart contract that represent possible vulnerabilities, and to model *inter-contract* interactions to check the robustness of a contract against some given attacks. Specifically, starting from the taxonomy of vulnerabilities causes in [3], we are building a catalog of:

– *patterns of properties* to guarantee the operational correctness of the contract and its adherence to certain predefined properties (i.e., intra-contract properties);
– *models of malicious contracts* to check the robustness of a contract against those vulnerabilities that allow malicious contracts to manipulate the behavior of the vulnerable contract (i.e., due to unsafe inter-contract interaction).

The long-term vision is to build an integrated environment that supports correct-by-design smart contract development, by providing automatic mappings between Solidity and ASMs, a GUI for ASM modeling, and seamless access to the verification back-end. There are also opportunities for extending this approach to other blockchain platforms beyond Ethereum. Overall, formal methods like ASMs can potentially increase the reliability and security of smart contracts, which is essential for their widespread adoption across domains.

References

1. Certora Technology White Paper. https://docs.certora.com/en/latest/docs/whitepaper/index.html, Accessed: 2024-02-20
2. Alchemy, N.: A short history of Smart Sontract hacks on Ethereum: A.k.a. why you need a smart contract security audit (2019)
3. Atzei, N., Bartoletti, M., Cimoli, T.: A Survey of Attacks on Ethereum Smart Contracts SoK. In: Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204. p. 164–186. Springer-Verlag (2017)
4. Börger, E., Raschke, A.: Modeling Companion for Software Practitioners. Springer (2018). https://doi.org/10.1007/978-3-662-56641-1
5. Börger, E., Stärk, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer Verlag (2003)
6. Braghin Chiara, Riccobene Elvinia, V.S.: State-based modeling and verification of smart contracts. Accepted to 39th ACM/SIGAPP Symposium on Applied Computing (2024)
7. Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC conference on Computer and Communications Security. pp. 254–269 (2016)
8. Madl, G., Bathen, L., Flores, G., Jadav, D.: Formal Verification of Smart Contracts Using Interface Automata. In: IEEE Int. Conf. on Blockchain. pp. 556–563 (2019)
9. Mavridou, A., Laszka, A.: Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach. In: Financial Cryptography and Data Security. pp. 523–540 (2018)
10. Permenev, A., Dimitrov, D., Tsankov, P., Drachsler-Cohen, D., Vechev, M.: Verx: Safety verification of smart contracts. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 1661–1677 (2020)
11. Tsankov, P., Dan, A., Drachsler-Cohen, D., Gervais, A., Bünzli, F., Vechev, M.: Securify: Practical security analysis of smart contracts. In: Proc. of Conference on Computer and Communications Security (CCS'18). p. 67–82 (2018)

[2] https://github.com/smart-contract-verification/ethereum-via-asm