Towards Compliance of Smart Contracts with the European Union Data Act

Luca Olivieri[1,†], Luca Pasetto[2,†]

[1]Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venice, Italy
[2]Department of Computer Science, University of Luxembourg, Esch-sur-Alzette,Luxembourg

Abstract
The recent proposal of the European Union Data Act adopted by the European Commission has the goal to create fair rules for sharing and using data. An important role in this context is the one played by smart contracts, that are considered as a key technology to enable effective and consensual data sharing. For this purpose, the proposal sets up some requirements for smart contracts, specifically in the context of interoperability with applications that process data. There is a need for scientific work that explains which steps to take so that smart contracts comply with the proposal. This paper aims to begin addressing this gap by considering each one of the requirements and by providing a series of techniques from different areas of computer science to comply with them.

Keywords
EU Data Act, smart contracts, blockchain, data sharing, interoperability, data privacy

1. Introduction

The recent proposal of the European Union Data Act has been published by the European Commission on 23 February 2022 [1]. Its goal is to create fair rules for sharing and using data in order to boost the EU's economy by making use of data generated from products and services. While there has been much discussion by scholars from different fields about the proposal itself, we believe that at this moment the scientific community is still lacking a work that considers which actions to take and computer science methods to adopt so that smart contracts are compliant with the requirements of the proposal. For instance, the article in [2] suggests recommendations on how to address smart contracts to improve the Data Act, but it does not analyze how existing technologies can be applied to satisfy the proposal. We plan to start filling this void with this paper.

In the proposal, a smart contract is defined as *"a computer program stored in an electronic ledger system wherein the outcome of the execution of the program is recorded on the electronic ledger"*. This definition differs from the historical one [3, Chapter 7] and it is closer to the concept of smart contracts for blockchain, even if this technology is not explicitly mentioned in the proposal. There can be different kinds of electronic ledgers, with blockchain being a *distributed*

OVERLAY 2023: 5th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 7, 2023, Rome, Italy
[†]These authors contributed equally.
luca.olivieri@unive.it (L. Olivieri); luca.pasetto@uni.lu (L. Pasetto)
0000-0001-8074-8980 (L. Olivieri); 0000-0003-1036-1718 (L. Pasetto)

one. It differs from traditional ledger systems because it does not require the involvement of third-party intermediaries, such as banks or notary companies. Instead, all transactions are securely recorded in a *transparent* and *immutable* way on a decentralized ledger and verified by network participants using cryptographic algorithms, reducing costs associated with traditional payment systems, as external intervention is not required.

All of the requirements for smart contracts set out in the Data Act can be found in Chapter VIII [1, Chapter 8], that treats interoperability, and more specifically in Article 28 and Article 30. Most of the requirements are actually in Article 30, which constrains data sharing involving smart contracts. On the other hand, Article 28 gives the essential requirements regarding interoperability, and it also pushes for interoperability of smart contracts within the services and activities of data space operators, especially in data sharing applications. This promotes seamless integration and execution of smart contracts across the data space ecosystem. Smart contracts are mentioned in Chapter VIII for a reason: besides the need for them to be regulated, they are seen in this proposal as a tool to increase interoperability in data sharing applications.

In this paper, we first describe the requirements for smart contracts and interoperability given by Article 30 and Article 28 of the EU Data Act proposal, and then we discuss state-of-art techniques to comply with each of these requirements.

2. Smart Contracts and Interoperability Requirements

In this Section, we shortly describe the requirements regarding smart contracts that are set out in Article 30 and Article 28 of the EU Data Act proposal. First, we focus on Article 30, which outlines responsibilities of the vendor of an application using smart contracts or, in case such a figure does not exist, the deployer of smart contracts for others. Following Article 30(1), they should comply with the following essential requirements:

1. *Robustness*: Ensure that smart contracts are highly robust to prevent functional errors and resist manipulation by third parties.
2. *Safe Termination and Interruption*: Implement mechanisms for terminating ongoing transactions within smart contracts, allowing for resetting or halting operations to prevent accidental executions.
3. *Data Archiving and Continuity*: Plan for archiving transactional data, smart contract logic, and code if the smart contract needs to be terminated or deactivated to maintain a record of past data operations (auditability).
4. *Access Control*: Implement rigorous access control mechanisms at both the governance and smart contract layers to protect smart contracts.

In addition to these requirements, the entire Chapter VIII deals with interoperability, and Article 28 states quite generically that *"the means to enable the interoperability of smart contracts within their services and activities shall be provided"*. Interoperability is defined in the proposal as *"the ability of two or more data spaces or communication networks, systems, products, applications or components to exchange and use data in order to perform their functions"*.

3. Discussion

Blockchain is a shared data structure with three main data properties: *immutability*, *distributability*, and *decentralization*. In a nutshell, the blockchain data structure (literally, a chain of blocks) is shared in redundant copies among a distributed and decentralized network and it allows to achieve that data is not tamperable (or rather, hardly tamperable) by a consensus mechanism and cryptographic primitives. Given its properties, by design, the blockchain is a promising data structure for smart contracts in view of satisfying the requirements of the EU Data Act.

We start by considering Article 30(1). For the first requirement, *robustness*, we recall that smart contracts become immutable after being deployed in the blockchain, and therefore they are resistant to manipulation by third parties. However, immutable programs, if not properly checked, may lead to errors, bugs, and vulnerabilities that are immutable as well. For this reason, the adoption of blockchain is not sufficient to completely guarantee robustness. In this context, static analysis can clearly be applied to detect issues in an early stage of code development and before code becomes immutable. Static analysis tools that check for errors in smart contracts already exist, see [4]. Besides proving the correctness of contracts by applying formal methods based on mathematical theories (see for instance [5], [6], [7]), specific extensions of these methods could be developed to prevent functional errors in the context of data sharing.

Regarding *safe termination* in the second requirement, we recall that deciding whether a Turing complete program terminates on all possible inputs is a famous undecidable problem in computer science [8]. However, blockchains exploit the *gas mechanism* to ensure termination of smart contracts: when a contract is executed, it also sets an amount of gas that is consumed during the execution of its instructions. If the gas is depleted before the execution is completed, then termination is forced, leading to a failure error. By applying formal verification, it is possible to prove the correct functioning of the gas mechanism [9] and to detect unexpected behaviors such as those due to *out-of-gas* vulnerabilities [10]. About the requirements for terminating an ongoing transaction, since each transaction is executed atomically, it is not possible to revert it after its approval. However, there are workarounds that permit to have code inside a contract to specify exceptions to its execution. For what regards *interruption* or reset operations to avoid accidental executions, there are already blockchain smart contracts that support these concepts, but there exist no standards yet. For example, the `selfdestruct` functionality of the Ethereum blockchain allows to destroy already deployed contracts [11]. Instead, the `Pausable` [12] contract proposed by OpenZeppelin implements an emergency stop mechanism that can be activated by an authorized account to avoid accidental executions.

Concerning the third requirement, *data archiving and continuity*, we have that a blockchain is based on a network where each peer keeps a (either full or partial) copy of the blockchain, and since this structure is distributed and decentralized, these copies are typically located in different geographical areas. In this manner, the blockchain keeps records of past data operations and avoids single points of failure. In addition, industrial blockchains, such as Hyperledger Fabric [13], allow to enrich contracts by importing general-purpose APIs to save data in different formats, media, and other backup technologies. Additionally, a blockchain might need to undergo maintenance operations, and it is essential that no data is lost during such phases: the situation differs between the cases of *permissioned* and *permissionless* blockchains. In the first case, access to the network and the ability to participate in the consensus process are restricted to a specific group of participants, and it is possible to perform operations through network governance. Moreover, this subset of peers also typically has the power to propose a plan for halting, modifying, and restarting the blockchain with updated software, carefully migrating the state of the previous version [14]. On the other hand, in a permissionless blockchain anyone can join the network, validate transactions, and participate in the consensus process without needing prior approval or identity verification. However, in this case maintenance can only be performed *on-chain*, as there is no high-level entity that governs the entire chain and each request must go through the consensus mechanism.

With respect to the fourth requirement, *access control*, we still need to consider the differences between a permissioned and a permissionless blockchains. In the first case, it is explicitly possible to set up governance on different levels, for instance by setting up a consortium [15] or private channels for the transmission of sensitive data [16]. In the second case, it is not possible to implement governance for the entire blockchain, as there are no permissions or restrictions to use the blockchain. However, it is always possible to do this at the smart contract layer: for instance, this is the case with contracts implementing Proxy Upgrade [17], which require to be managed by special users named *contract admin*. In addition, blockchain technology can be supported by models of self-sovereign identity [18] for digital identity management, where users can control and use their sensitive data without disclosing it in the blockchain, thanks to zero-knowledge protocols. Moreover, anomaly detection for blockchain [19] can be applied to detect user behaviors that fall outside the normal range also after access has been granted.

The last point is the one about *interoperability*, which is the core of Chapter VIII and of Article 28 [1]. As intuition suggests, an increase in the interoperability of smart contracts poses new challenges, both technical for data sharing and ethico-legal for data privacy. We identified two points of discussion regarding this. First, there is a problem of interoperability in-between blockchains, as there is a lack of standardization for smart contracts, and different blockchain platforms use different programming languages and even different consensus mechanisms. Moreover, different blockchains may operate under different legal and regulatory frameworks, and interoperability solutions need to account for these differences to ensure compliance. Having a common language might not be achievable, but there is a need for at least a cross-language framework for different blockchains to interact. However, such a framework implies new non-trivial verification challenges [20]. Second, blockchains need to be interoperable also with external data and applications [21]: in order for smart contracts to be useful, they often need to be parameterized, i.e., to change their behavior depending on external data sources. This can be achieved by using *oracles* that provide data to a smart contract. However, it might be difficult to decide whether an oracle is trustable and reliable. An approach to this problem is to use symbolic methods from the artificial intelligence research area: while subsymbolic methods typically provide good results for a variety of tasks, they oftentimes are not *explainable*, that is, they do not provide an explanation for their decisions. For instance, [22] gives a framework that incorporates formal argumentation and negotiation in a blockchain environment to make the decision-making process of an oracle transparent. After having the information of which are the trustable oracles, we need to track the data in the contract to the corresponding oracle source. Program analysis can help in tracking the information flows inside smart contracts, and, for instance, it has already been used to identify data leaks related to GDPR in traditional programs [23, 24, 25].

4. Acknowledgments

Luca Olivieri: work partially supported by SERICS (PE00000014) under the NRRP MUR program funded by the EU and SPIN-2021 "Ressa-Rob" funded by Ca' Foscari University of Venice.

References

[1] European Commission, Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on harmonised rules on fair access to and use of data (Data Act), COM/2022/68 final, 2022. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2022:68:FIN.

[2] F. Casolari, M. Taddeo, A. Turillazzi, L. Floridi, How to improve smart contracts in the European Union Data Act, Digital Society 2 (2023) 9. doi:10.1007/s44206-023-00038-2.

[3] A. M. Antonopoulos, G. Wood, Mastering Ethereum: Building Smart Contracts and Dapps, O'Reilly, 2018.

[4] P. Tolmach, Y. Li, S.-W. Lin, Y. Liu, Z. Li, A survey of smart contract formal specification and verification, ACM Comput. Surv. 54 (2021).

[5] C. Patrick, Principles of Abstract Interpretation, MIT Press Academic, Cambridge, MA, USA, 2021.

[6] E. M. Clarke, Jr., O. Grumberg, D. A. Peled, Model Checking, MIT Press, Cambridge, MA, USA, 1999.

[7] J. H. Gallier, Logic for computer science: foundations of automatic theorem proving, Courier Dover Publications, Mineola, NY, USA, 2015.

[8] H. G. Rice, Classes of Recursively Enumerable Sets and Their Decision Problems, Transactions of the American Mathemathical Society 74 (1953) 358–366. doi:10.1090/s0002-9947-1953-0053041-6.

[9] T. Genet., T. Jensen., J. Sauvage., Termination of ethereum's smart contracts, in: Proceedings of the 17th International Joint Conference on e-Business and Telecommunications - SECRYPT, INSTICC, SciTePress, 2020, pp. 39–51. doi:10.5220/0009564100390051.

[10] E. Albert, J. Correas, P. Gordillo, G. Román-Díez, A. Rubio, Gasol: Gas analysis and optimization for ethereum smart contracts, in: A. Biere, D. Parker (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer International Publishing, Cham, 2020, pp. 118–125.

[11] J. Chen, X. Xia, D. Lo, J. Grundy, Why do smart contracts self-destruct? investigating the selfdestruct function on ethereum, ACM Trans. Softw. Eng. Methodol. 31 (2021). doi:10.1145/3488245.

[12] OpenZeppelin, Pausable, 2023. https://docs.openzeppelin.com/contracts/2.x/api/lifecycle#pausable (Accessed 09/2023).

[13] Hyperledger, Hyperledger Fabric Documentation, 2023. https://hyperledger-fabric.readthedocs.io/en/release-2.5/whatis.html (Accessed 09/2023).

[14] L. Olivieri, F. Tagliaferro, V. Arceri, M. Ruaro, L. Negrini, A. Cortesi, P. Ferrara, F. Spoto, E. Talin, Ensuring determinism in blockchain software with golisa: an industrial experience report, in: L. Gonnord, L. Titolo (Eds.), SOAP '22: 11th ACM SIGPLAN International

Workshop on the State Of the Art in Program Analysis, San Diego, CA, USA, 14 June 2022, ACM, 2022, pp. 23–29. URL: https://doi.org/10.1145/3520313.3534658. doi:10.1145/3520313.3534658.

[15] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, E. B. Hamida, Consortium blockchains: Overview, applications and challenges, Int. J. Adv. Telecommun 11 (2018) 51–64.

[16] Hyperledger, Channels, 2023. https://hyperledger-fabric.readthedocs.io/en/release-2.5/channels.html (Accessed 09/2023).

[17] OpenZeppelin, Proxy Upgrade Pattern, 2023. https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies (Accessed 09/2023).

[18] A. Preukschat, D. Reed, Self-sovereign identity, Manning Publications, 2021.

[19] M. Ul Hassan, M. H. Rehmani, J. Chen, Anomaly detection in blockchain networks: A comprehensive survey, IEEE Communications Surveys & Tutorials 25 (2023) 289–318. doi:10.1109/COMST.2022.3205643.

[20] L. Negrini, P. Ferrara, V. Arceri, A. Cortesi, LiSA: A Generic Framework for Multi-language Static Analysis, Springer Nature Singapore, 2023, pp. 19–42. doi:10.1007/978-981-19-9601-6_2.

[21] S. Porru, A. Pinna, M. Marchesi, R. Tonelli, Blockchain-oriented software engineering: Challenges and new directions, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 2017, pp. 169–171. doi:10.1109/ICSE-C.2017.142.

[22] L. Yu, M. Zichichi, R. Markovich, A. Najjar, Enhancing trust in trust services: Towards an intelligent human-input-based blockchain oracle (ihibo), in: Proceedings of the 55th Annual Hawaii International Conference on System Sciences, CE - Commission Européenne [BE], January 2022. doi:10.5220/0010945300003116.

[23] P. Ferrara, L. Olivieri, F. Spoto, Static Privacy Analysis by Flow Reconstruction of Tainted data, Int. J. Softw. Eng. Knowl. Eng. 31 (2021) 973–1016. doi:10.1142/S0218194021500303.

[24] P. Ferrara, F. Spoto, Static analysis for GDPR compliance, in: CEUR Workshop Proceedings - Proceedings of ITASEC '18, volume 2058, 2018, pp. 1–10.

[25] P. Ferrara, L. Olivieri, F. Spoto, Tailoring Taint Analysis to GDPR, in: Privacy Technologies and Policy - 6th Annual Privacy Forum, APF 2018, Barcelona, Spain, June 13-14, 2018, Revised Selected Papers, volume 11079 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 63–76. doi:10.1007/978-3-030-02547-2_4.