Standardizing Smart Contracts

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 August 2022, accepted 19 August 2022, date of publication 29 August 2022, date of current version 2 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3202550

RESEARCH ARTICLE

Standardizing Smart Contracts

VITTORIO CAPOCASALE[1], (Member, IEEE), AND GUIDO PERBOLI[2], (Member, IEEE)
[1]Department of Control and Computer Engineering, Polytechnic University of Turin, 10129 Turin, Italy
[2]Department of Management and Production Engineering, Polytechnic University of Turin, 10129 Turin, Italy

Corresponding author: Vittorio Capocasale (vittorio.capocasale@polito.it)

This work was supported by the Project Cyber security cOmpeteNCe fOr Research anD InnovAtion (CONCORDIA) of the European Union (EU) Commission, under Grant 830927.

ABSTRACT In the evolving context of distributed ledger technologies, the standardization of smart contracts is necessary. Smart contracts are tamper-proof computer programs. Due to their security and flexibility, it is possible to exploit smart contracts in a wide variety of use cases. In particular, it could be possible to automate legally recognized contracts by leveraging smart contracts. To this extent, some standards regarding the proper management of smart contracts are surging. However, there are still many technological misconceptions regarding smart contracts. This study describes smart contracts from multiple perspectives and identifies and clarifies some of the most common misconceptions regarding smart contracts. This study also provides some guidelines and insights on the proper management of smart contracts. This study can be a valuable resource for future standards on smart contracts.

INDEX TERMS Blockchain, chaincode, smart contracts, standardization.

I. INTRODUCTION

Blockchain technology is revolutionizing the world. Decentralized versions of well-established centralized services are surging in multiple areas, like finance [1], insurance [2], logistics [3], [4], energy [5], [6], and more, pushing the digitalization of the physical world to new frontiers. Digital tokens are gaining adoption: NFTs and exchangeable coins are increasing their market share [7], [8] as they allow tokenizing and trading even historically illiquid assets (e.g., mining power [9]).

All these transformations are mainly possible because of the flexibility of smart contracts [10]. Smart contracts are tamper-proof computer programs, as the security of the blockchain technology guarantees the correctness of their execution. Thus, smart contracts could automate and enhance the fairness of critical processes, guarantee the quality of data sources, and protect valuable resources, which are topics of relevant interest [11], [12], [13]. In particular, the idea of automating legally recognized contracts is very appealing [14].

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

However, creating tamper-proof, secure, decentralized, economically advantageous, and legally recognized smart contracts is not straightforward: it is necessary to consider various aspects, which are both technical and law-related [15]. In particular, smart contracts are rarely used as a standalone technology. In logistics, for example, drones automate on-field operations [16], [17], and IoT devices collect on-field data to be elaborated through artificial intelligence, stochastic programming, and granular computing approaches [4], [18], [19], [20]. Thus, to guarantee smart contracts use sufficiently decentralized data feeds, a single product should be tracked by multiple IoT devices. Nonetheless, this approach is often inconvenient due to economic or physical constraints. Moreover, processing data from redundant sources increases the complexity of the data elaboration algorithms without improving their accuracy. Overall, establishing a comprehensive smart contract framework is challenging as multiple concerns must be addressed simultaneously. Thus, standards appeared in the literature only recently [21].

Unsurprisingly, there are still many misconceptions regarding smart contracts. Smart contracts are hard to frame, even from a purely technical standpoint: many blockchain platforms provide functionalities related to smart contracts but

with different deployment and execution strategies which is hard to comprise under an unified definition. The coupling of this variety with the increasing interest in the topic, in particular from people with different backgrounds and priorities, has created the perfect environment for the growth of several misunderstandings and partial truths. The smart contract name itself is misleading, as it hints at legally recognized digital agreements, not at general-purpose computer programs. Moreover, many properties of smart contracts depend on the configuration and degree of decentralization of the underlying blockchain system, which often makes generalization attempts inaccurate. In such a context, the creation of proper definitions, standards, guidelines, and best practices for smart contracts is becoming an impelling necessity. Additionally, given the broad audience interested in the topic, smart contracts must be discussed in a precise, correct, and widely understandable manner, which makes the task even more challenging.

The study aims to create a common ground for future standards on smart contracts. This study can be helpful to researchers, decision-makers, lawyers, and computer scientists in understanding the potential and caveats related to smart contracts. In particular, this study provides the following contributions:

- it describes smart contracts from multiple perspectives to provide a clear overview even to non-technical readers;
- it identifies and corrects some common misconceptions related to smart contracts;
- it provides some guidelines on the proper implementation and execution of smart contracts to underline the potential, the trade-offs, and the limits of the technology. The objective of the proposed guidelines is to set the goals that future standards should aim for, not to describe some strict rules. Thus, we expect future standards to adhere to the proposed guidelines as much as possible, as long as it is reasonable.

The remaining part of this study is structured as follows: Section II briefly describes the smart contract concept, the blockchain technology, and presents a literature review; Section III analyzes smart contracts; Section IV concludes the study.

II. BACKGROUND

This section summarizes the main concepts related to blockchain and smart contracts. Moreover, this section includes a short literature review.

A. BLOCKCHAIN

Blockchain is a technology that enables the sharing of data among non-trusting parties [22], as it allows for solving trust issues among non-trusting parties without leveraging any trusted third one [23]. Blockchain is composed of a network of nodes that share a common database [24]. The shared database has the structure of a ledger: data can only be added to it. Each node has its copy of the ledger, and each node manages its copy independently of the other nodes. Consequently, while each node can arbitrarily modify its copy, the global state of the ledger is established based on what the majority of the copies store [25]. Thus, the state of a blockchain system is updated based on majority voting. In this work, we assume a uniform distribution of voting power among the nodes to simplify the discussion. However, when we refer to the majority of the peers, we actually mean the majority of the voting power.

B. SMART CONTRACTS

Smart contracts are not bonded to blockchain technology, and their original definition dealt with the automation of legal contracts [26]. However, in the context of blockchain (and other distributed ledger technologies), smart contracts have assumed a different meaning: they are tamper-proof computer programs that update the state of the ledger [27]. In fact, by running the code of a smart contract on multiple nodes, the probability of successfully altering its execution is negligible. Moreover, smart contracts can execute arbitrary logic, which makes them usable for automating tasks of different nature by expressing execution conditions and reacting to events (generated by users or other smart contracts). This study will limit its analysis to smart contracts that are used in blockchain systems. Section III analyzes smart contracts in depth.

C. PROBLEM STATEMENT

Currently, research on smart contracts is moving fast in multiple fields. Researchers struggle to understand each other due to their different backgrounds and perspectives, including computer science, economy, and law. Often, important details are neglected to provide understandable descriptions of the smart contract topic, which has created the perfect environment for the growth of several misunderstandings and partial truths. Even emerging standards on smart contracts are affected by such limitations.

This study identifies some of the most common misconceptions and provides a high-level description of smart contracts. We discuss the topic without neglecting the necessary technical details, which allows us to provide some guidelines that could be helpful to both technical and non-technical readers to demystify the technology and understand its limits and where it could find adoption.

D. LITERATURE REVIEW

Many authors dealt with smart contracts in the literature, particularly in the last decade. However, this study analyzes the topic from a unique perspective, as it provides both practical guidelines and philosophical interpretations.

Smart contracts were introduced to digitalize and automate legal contracts [26]. The term has then been adopted in the blockchain context to identify code scripts executed by the nodes of a blockchain network [27]. Thus, smart contracts identify two different concepts [15], [28]. In particular, a study distinguished between smart contract code

(which is executed in blockchain systems) and smart legal contracts, which are legal contracts in digital form [15]. The study analyzed smart contracts mainly from a legal perspective and underlined how smart contract code is just a portion of smart legal contracts [15]. Another study argues that there are only small advantages to using blockchain-based smart contracts from a practical standpoint [29], as many terms indicating the properties of blockchain-based smart contracts are often misleading and not applicable in a legal context [29].

Such ambiguities and misconceptions have significantly slowed down the creation of standards for smart contracts [30]. Some authors identified the main issues and limitations in the automation of contracts from a legal perspective (e.g., the definition of the scope of smart contracts, their internationalization, their applicability, and their validity) [31]. Others provided the essential requirements that smart legal agreements must meet, focusing on bridging the gap between smart legal contracts and smart contract code and streamlining such a process by redefining existing standards [32]. Other studies outlined the key parameters to consider in ensuring the legal recognition of smart contracts [33]. In particular, the definition of universal APIs, coding standards, and conflict resolution mechanisms are essential [34]. Instead, this work aims to demystify and standardize blockchain-based smart contracts from a computer science perspective, with a focus on the application-level implications of the provided guidelines.

Some authors identified some of the main characteristics of smart contracts (e.g., transparency, availability, and immutability) [35], others proposed new designs to address some of the current shortcomings (e.g., transaction order dependency, determinism, and exception management) [36] and full standards can be found in the gray literature [21]. However, such works are biased towards the vision of smart contracts introduced by the Ethereum protocol [27]. Moreover, some of the guidelines proposed in [21] should be revised. For example, the use of timers to terminate contracts may cause inconsistencies at execution or verification time, as discussed in Sec. III-C.

Some studies proposed standards for some specific use cases. For example, guidelines for financial smart contracts [37] and for altering and undoing smart contracts [38] are available in the literature.

Other works focused on the paradigms and tools related to smart contracts [28], [39], including strategies to reduce gas fees [40]. Some authors provided a formal description of smart contracts and their characteristics [41], and others described the issues of programming smart contracts and proposed some possible solutions [42], [43]. Such studies, however, have a strong focus on purely technical perspectives.

III. SMART CONTRACTS: MISCONCEPTIONS AND GUIDELINES

This section identifies some common misconceptions related to blockchain-based smart contracts and provides guidelines for their standardization. Table 1 summarizes the contents of this section.

A. SMART CONTRACTS ARE STATE-TRANSITION FUNCTIONS

It is possible to describe a blockchain system as a finite state machine [27]. Under this perspective, smart contracts are the state-transition functions that move the system from one state to the other: it is possible to describe a smart contract as a function $\delta\colon \mathcal{S}\times\mathcal{I}\to\mathcal{S}$, where $\mathcal{S}$ is the finite and non-empty set of the states of the blockchain ledger, and $\mathcal{I}$ is the set of the possible input transactions.

From a broader perspective, this guideline states that smart contracts operate on two types of data: the (internal) ledger data, which is reliable, and the (external) transaction data, whose correctness must be verified. Depending on the use case, ledger data may not be enough to verify transaction data. Thus, there is an intrinsic limit on the usability of smart contracts. Overcoming such a limit by accepting partially-verified transaction data is possible, but such an approach introduces the garbage in, garbage out problem (i.e., ledger data is no longer reliable).

B. SMART CONTRACTS MUST ALTER THE STATE OF THE SYSTEM

Different from generic state-transition functions, a smart contract must alter the state of the system: if a smart contract only reads some data, or if it only does some data elaboration without altering the system's state, it becomes impossible to check the correctness of the execution of the smart contract. For example, the Ethereum protocol allows for the definition of pure or view functions [44], as shown in the following code snippet:

```
contract Logger {
  uint64 rowID;
  function persist(uint64 id) external {
    rowID=id;
  }
  function lastID() view external
          returns (uint64) {
    return rowID;
  }
}
```

When pure or view functions are invoked, the execution takes place on a single node. Consequently, the node could provide a wrong response. Nonetheless, this is the expected behavior: pure or view functions are not smart contracts but only commodity functions for fetching data from the Ethereum blockchain or performing computations on transaction data only. Considering pure or view functions like smart contracts is a common misconception [45], [46], probably caused by the fact that their declaration occurs inside a data structure identified by the keyword *contract*. Other platforms do not explicitly differentiate among functions based on their interaction with the ledger, but the general concept still

TABLE 1. An overview of smart contracts. The table presents some categories. The table summarizes the main claims presented in this work for each category.

Category	Claims: a smart contract...
Perspectives	is a computer program is a state-transition function is an equivalence class belongs to the system
Properties	is digital is likely tamper-proof is likely bug-free
Requirements	must be verifiable must be deterministic must use blockchain to store its output
Misconceptions	is immutable has to be stored on-chain has to be certified might not alter the state of the system has legal value has an intrinsic meaning and interpretation
Guidelines	should be independently coded (preferably) should be independently tested (preferably) should be independently executed (preferably) should be defined by what it does (not by how) should be avoided if the code must be certified should rely on oracles as little as possible should not rely on external data sources (e.g., the node's clock) should not rely on undefined behaviors (e.g., iteration order) should provide access to its source code should be integrated into a framework that defines its meaning may be integrated into a framework that grants it some legal value may leverage execution proofs

applies: the only verifiable output of a smart contract is the one that is stored in the blockchain. The reason is that it is possible to verify if peers reach consensus on a write but not on a read operation: a malicious peer could answer a query even if it should not by only leveraging its copy of the ledger, but it cannot force a write operation without altering the majority of the copies.

This guideline has relevant practical implications. For example, when generating a certificate (or performing any other read-only operation), the smart contract should also store the generated certificate (or its fingerprint) in the blockchain. When reading the certificate (or any data from the blockchain), it is not possible to leverage smart contracts, and a sufficient number of nodes should be queried. As this is often impractical, users are likely to trust a few reputable nodes. Thus, a certain amount of trust and centralization is still present in blockchain systems.

C. SMART CONTRACTS MUST BE VERIFIABLE

It should be possible to verify the output of a smart contract at any given point in the future: if this requirement is not met, the system may fork, and consensus may never be reached. In particular, the time of verification can be very different from the time of execution of a smart contract, which introduces a limitation on the use of time-related primitives [47]: even if all the nodes in the system share the same atomic clock, this would only allow them to synchronize the execution of a given smart contract, but not its verification. Such a limitation does not prevent smart contracts from leveraging time-related primitives, but their introduction should be pondered carefully. For example, checking if the timestamps of two blocks are less than a month apart will return the same value at any given point in the future. On the other hand, performing the same check with the timestamp of the last block is problematic, as its result will change with time. Thus, time-based smart contracts such as those proposed in [21] can cause inconsistencies at verification time. Consequently, smart contracts should only rely on the data already stored in the blockchain or provided within the input transaction. Smart contracts should not rely on anything else (e.g., the time measured by the node executing the smart contract).

This guideline has important implications from both a managerial and a legal standpoint, as some use cases may be problematic to model through smart contracts: it is possible to determine if a given event happened in a certain time window but not the exact moment. In Bitcoin [48], for example, block validity checks include countermeasures to prevent miners from manipulating the timestamp of blocks. Such countermeasures, however, still allow for a two hours time wiggle. Thus, deadlines can only be approximately checked.

D. SMART CONTRACTS MUST BE DETERMINISTIC

The previous section introduces a more general key point. A smart contract must produce the same output on all the nodes executing it, not only at any given point in the future. In other words, a smart contract must be deterministic. This requirement is often underestimated and is not limited to time-based primitives. For example, maps are unordered sets

of key-value pairs in the Go programming language. Thus, the iteration order of a program looping through a map is not guaranteed. This behavior can affect serialization libraries as well. More generally, randomness in smart contracts must be managed carefully. This problem is actively researched, and some of the main techniques to address it are based on multi-party computation [49].

From a broader perspective, smart contracts require the standardization of the representation and processing of data, as this allows different peers to reach the same state.

E. SMART CONTRACTS ARE EQUIVALENCE CLASSES

A common misconception is that a smart contract is unique [50]. However, in blockchain systems, a node cannot check which computations are performed by the others but only if they reach the same state after the computation: different transition functions that produce the same resulting state are indistinguishable in a blockchain context. Thus, a smart contract is a class of equivalent state transition functions that, when given the same input state and input symbol, produce the same output state. For example, the following code snippet shows two equivalent implementations of the same smart contract:

```
func sum(a uint8, b~uint8) {
  a = a + b
  store(a)
}
func sum(a uint8, b~uint8) {
  var i uint8
  for i = 0; i < b; i++ {
    a = a + 1
  }
  store(a)
 }
```

A blockchain system works without issues even if some nodes use the first function while the others use the second one. Empirical proof of such a statement is available on Github [51]. This observation is particularly relevant when smart contracts must have legal value, as it would be necessary to define what smart contracts should do and not how they should do it [52]. Thus, smart contracts should be expressed in declarative languages. The creation of such languages is an active research field, and it would make smart contracts easier to implement and understand [52].

F. SMART CONTRACTS DO NOT NEED TO BE STORED ON-CHAIN

Another common misconception, probably introduced by the Ethereum protocol, is that smart contracts have to be stored on-chain [53], [54]. As discussed in the previous section, a node in a blockchain system cannot check which computation the other nodes perform. Consequently, storing the code of a smart contract on-chain is not a way to force all the nodes to use the same implementation. Storing the code of a smart contract on-chain is only a simple way to distribute the smart contract code to all the blockchain nodes. Public blockchains (e.g., Ethereum) use this approach to automatically update the set of smart contracts deployed on each node. Other platforms (e.g., Sawtooth [55]) do not store the code on-chain, and each peer is responsible for installing the required smart contracts on its node. The key idea is that each node does not need to know what code the others are running, but just which state they reach in the end. Assuming that the majority is honest, a common state will be reached. Empirical proof of this guideline is available on Github [51].

Storing the code of a smart contract on-chain could be valuable in other contexts. For example, it could be important to track which version of a smart contract was running at a given point in the past for verifiability purposes. Moreover, the on-chain version could be the one that a tribunal should use in case of litigation. However, if the on-chain implementation is the only one having legal value, the peer that codes it could manipulate the behavior of the smart contract for selfish purposes (e.g., by hiding a backdoor). Even if the honest majority forks the system to restore its correct state, a tribunal could be forced to overrule the majority-based decision of the blockchain network and favor the legally recognized branch generated by the malicious peer. Thus, if the on-chain implementation is the only one having legal value, additional strategies must be implemented to guarantee its correctness (e.g., formal approval from the majority of peers).

G. SMART CONTRACTS ARE NOT IMMUTABLE

This misconception is related to the previous one. If a blockchain is immutable, and a smart contract is stored on-chain, then the smart contract is immutable [56], [57], [58]. However, the term *immutable* is misleading. More correctly, a blockchain is append-only. Consequently, even if what is stored in a blockchain cannot be altered, a newer version of it can always be appended to the ledger. Thus, even when smart contracts are stored on-chain, they can be updated. One possible strategy for the Ethereum blockchain is described in [59]: the smart contract stores the address of another smart contract in one of its state variables. Whenever the original smart contract is invoked, it propagates the invocation to the smart contract at the stored address. The behavior of the original smart contract is modifiable by updating the stored address. Additionally, the peer majority can always decide to soft/hard fork the system to replace a given smart contract. Thus, Ethereum smart contracts are immutable only if they do not implement an update mechanism and the majority is unwilling to alter them. Thus, Ethereum smart contracts are tamper-resistant, more than immutable. Other platforms (e.g., EOSIO [60]) allow updating smart contracts by overwriting the old code with the new one [61].

Unfortunately, updating smart contracts is not easy, as all the nodes should start using the newer version simultaneously to avoid partitioning the system. As each node acts independently from the others, smart contract updates cannot be enforced like in centralized systems but must be proposed and accepted by the peers. The acceptance strategy depends

on the platform and, among others, may require verifying the correctness of the update transaction (e.g., by checking that the creator of the original smart contract is also the updater) or collecting the explicit approval of the peers. Thus, managing smart contracts is more challenging than managing centralized programs, as it requires the cooperation and coordination of the majority of the nodes.

H. SMART CONTRACTS ARE NOT LEGAL CONTRACTS

As a consequence of their name, blockchain-based smart contracts are often considered to be contracts [62], [63], [64], which is untrue [10]: blockchain-based smart contracts are computer programs. Consequently, they have an enormous range of possible applications and can represent much more than an agreement [10]. From this perspective, smart contracts are more than just legal contracts. Nonetheless, smart contracts do not usually declare who should use them, and users do not digitally sign smart contracts but the transactions to interact with them. Thus, smart contracts may not be enough to be legal contracts, and their legal value must be legitimized by existing legal tools. In some jurisdictions, the definition of a legal contract may already apply to smart contracts; in other jurisdictions, it may be necessary to legitimize smart contracts with parallel legal contracts [65]. In any case, smart contracts could automate legal contracts, at least in part [29]. However, this is an open research field and requires the careful design of a hybrid framework comprising both computer science and law-related aspects [15], [66].

In the general case, a standalone blockchain cannot replace existing contracts or be considered as the source of undeniable evidence in litigation. Smart contracts can standardize and simplify data sharing across multiple companies, but their legal legitimization is only potential and depends on the jurisdiction. Managers should carefully consider the economic implications of the adoption of blockchain technology without assuming for granted a reduction of the legal costs.

I. SMART CONTRACTS DO NOT PROVIDE MEANING

Smart contracts are computer programs: they elaborate sequences of bits and produce other sequences of bits. However, smart contracts do not detail the meaning and the correct interpretation of the sequences of bits that they produce. For example, a smart contract may store in blockchain the following sequence: *e*, *s*, *t*, *a*, *t*, *e*. While it is possible to assume that the sequence represents the word *estate*, this cannot be guaranteed. The letters could be the result of a random extraction. In such a case, they should be interpreted separately. Moreover, the word *estate* assumes different meanings in Italian and English (it is a false friend). Consequently, different interpretations may arise from the same data. Thus, smart contracts (by themselves) are not enough: they need external standards to establish how data should be encoded/decoded and how data consumers should interpret it. The rules for data interpretation cannot be naively stored in blockchain, as the problem would become circular.

To some extent, all blockchain protocols (e.g., Ethereum) implicitly define data encoding and decoding standards. However, as they try to be agnostic and general purpose, they do not provide enough details to guarantee a univocal and meaningful interpretation of the stored data, which is required for legitimizing smart contracts from a legal standpoint. From a managerial perspective, the rules for data interpretation can be implicit, and no external framework is needed if smart contracts are only used to share data among multiple companies. However, to have legal value, such rules must be explicit (or, somehow, commonly recognized and acknowledged), and smart contracts should be integrated into a hybrid framework comprising both computer science and law-related aspects.

J. PREFERABLY, SMART CONTRACTS SHOULD BE INDEPENDENTLY CODED AND DEPLOYED

In opposition to what many blockchain networks currently do (e.g., Ethereum), smart contracts should not be coded once and deployed on all the nodes of the system, as this is contrary to the intrinsic idea of blockchain, which is a system where nodes do not trust each other. Consequently, a node should never accept to execute the implementation provided by other untrusted nodes. Instead, each node should autonomously implement all the smart contracts to be certain of their correctness. As long as all the honest nodes share a common vision on how the smart contracts should alter the system's state, all the independent implementations should belong to the same equivalence class. Thus, all the nodes can reach the same state, even if they use different implementations of the same smart contract.

Unfortunately, requiring that each node implements its smart contract is often unfeasible. There are some use cases where a few actors need a consortium blockchain with a few smart contracts that all of them use (e.g., in logistics [67], [68]). In such cases, each actor can afford to code and deploy its implementation of the smart contract. However, in public blockchains, nodes usually use only a portion of the totality of the smart contracts available. Moreover, they often lack the proper competencies and resources to create independent implementations. In such situations, requiring that each node independently codes all the smart contracts (even those that it does not use) is too demanding. Nonetheless, maliciously exploiting a bug becomes almost impossible with just a few independent and uniformly distributed implementations, as the bug would likely affect only a portion of the network. Thus, incentivizing the creation of multiple implementations and allowing peers to choose which one to install on their node could reduce the risk of bug exploits in smart contracts.

From a managerial standpoint, this guideline has a relevant impact on the economy of blockchain-based projects, as the development costs are not shared but replicated across the various nodes. Moreover, additional efforts are required to ensure that all the independent implementations belong to the same equivalence class. In particular, the creators of widely

used smart contracts could fund the creation of independent implementations to offset the risk of bug exploits. Similar initiatives already exist in the form of bug bounty programs.

K. PREFERABLY, SMART CONTRACTS SHOULD BE INDEPENDENTLY AUDITED AND TESTED

Smart contracts should be independently audited and tested by the nodes before they start using them. Thus, similar considerations to the ones made for the previous guideline still apply. However, this guideline is likely to find more adoption, as testing a smart contract should be easier than coding it. In order to streamline the testing process, the publication of the source code of smart contracts is a best practice, as it simplifies the auditing of the machine code.

From a practical standpoint, nodes are unlikely to test smart contracts for the same lack of resources and competencies discussed in the previous guideline, in particular in public blockchain networks. Thus, particularly when economic incentives for testing are lacking, smart contracts may not be as secure and trustworthy as they are often considered.

L. SMART CONTRACTS MAY LEVERAGE EXECUTION PROOFS

In some situations, requiring independent executions of computationally-heavy smart contracts may be too demanding, and leveraging a single execution and the proof of its correctness may be preferable. For example, finding a solution to the Rubik's cube may be difficult. Nonetheless, if the moves are provided, it is simple to check whether or not they compose a solution. The following code snippet shows the two alternative approaches:

```
func rubik_heavy(problem Problem) {
  solution := solve(problem)
  if solution != nil {
    store(solution)
  }
}
func rubik_light(problem Problem,
                 moves []Move) {
  solution := verify(problem, moves)
  if solution != nil {
    store(solution)
  }
}
```

Interestingly, the lightweight approach still requires executing a portion of the protocol as a smart contract (i.e., the solution verification and storage). Thus, smart contracts have an intrinsic complexity baseline, as the verification and storage steps must always be performed in a decentralized fashion. Nonetheless, many blockchain platforms are successfully adopting execution proof to improve scalability. To this extent, zero-knowledge proofs are particularly helpful [69] and are currently used in protocols like zkSync [70].

M. SMART CONTRACTS DO NOT NEED TO BE CERTIFIED

Independently of the use case, smart contracts do not need to be certified: certification bodies are trusted third parties, and blockchain technology tries to eliminate all such parties. Of course, it is in the best interests of the nodes to properly test and audit their smart contract implementations [71], which can also include relying on external software testing services. However, a blockchain system should not recognize only a few authorities to validate smart contracts. The correctness of smart contracts should be guaranteed only by the fact that the nodes can reach the same state. If feasible, each node should code and test its implementation according to the strategies that the node considers appropriate. In case a blockchain system wants to rely on some external authorities to certify smart contracts, the following strategies are better alternatives:

- switch to a centralized system controlled by the certification body. In fact, by leaving a backdoor in a smart contract, the certification body would have total control over the smart contract all the same;
- use oracles. Oracles are trusted third parties that provide data coming from the real world to the blockchain system. Thus, oracles are often necessary, as real-world data can not be (reasonably) obtained otherwise. If a smart contract has to be certified, it would be simpler and more efficient to let the certification bodies run the code and store the result in the blockchain. By using a decentralized oracle network [72], the influence of each oracle can be limited. In this way, it is possible to check the coherence of the results produced by the various certification bodies. The problem with this solution is that the decentralization degree is proportional to the number of certification bodies and not to the number of nodes.

From a broader perspective, any form of centralization undermines the value of blockchain-based smart contracts. Thus, before settling for blockchain-based solutions, decision-makers should consider to which extent it is possible to decentralize a system. In many situations, the impracticality of decentralized solutions may drive decision-makers to accept centralized compromises in their blockchain systems. However, this approach may defeat the purpose of having a blockchain in the first place.

N. PREFERABLY, SMART CONTRACTS SHOULD NOT RELY ON ORACLES

The problem of relying on oracles is not exclusive to the code certification process. It is never possible to fully trust data provided by oracles: an oracle is always a trusted third party that reduces the degree of decentralization of the system [29]. Unfortunately, eliminating oracles from a blockchain system is rarely feasible. Thus, smart contracts should rely on data provided by oracles as little as possible. Moreover, strategies that discourage oracles' misconduct should occur. To this extent, oracle services (e.g., Chainlink [72]) use economic

incentives and multiple data feeds to reduce manipulation attempts.

More generally, smart contracts do not solve the garbage in, garbage out problem when oracles are involved. Thus, blockchain can be used to prevent retroactive (but not proactive) data manipulations. Based on the specific use case, decision-makers should evaluate whether such guarantees justify the adoption of the technology or not.

O. SMART CONTRACTS ARE (LIKELY) BUG-FREE

Under the assumption that a smart contract is independently coded by multiple nodes, it is unlikely that all the implementations share the same bug. Of course, all the implementations might rely on the same library. In such cases, bugs affecting the library would also affect all the implementations. However, the general idea behind the scalability trilemma [73] still applies: increasing the number of independent implementations increases the effort required to code them and the probability of obtaining a bug-free smart contract, as already discussed in Sec. III-J.

We underline that it is likely possible to obtain similar bug-free programs in centralized settings by sustaining similar implementation efforts. As this is not the standard industrial behavior, the cost-effectiveness of such a strategy is questionable. Nonetheless, we believe that smart contracts that manage valuable assets (e.g., protocols for decentralized finance) could consider this strategy as a way to offset the risk of cyberattacks.

P. SMART CONTRACTS ARE (LIKELY) TAMPER-PROOF

As a smart contract is independently executed/verified by multiple nodes, corrupting the execution of a smart contract is reasonably unfeasible. Consequently, smart contracts are tamper-proof, but only as long as the nodes do not have a motivation to collude. Moreover, the tamper-proof property does not imply that the smart contract behaves as expected: the tamper-proof property is a consequence of the multiple independent executions of a smart contract, while its correctness is a consequence of its multiple and independent implementations. Consequently, even if all the nodes used identical implementations, the smart contract would still be tamper-proof (but unlikely bug-free).

We underline that the tamper-proof resiliency of smart contracts is proportional to the decentralization degree of the blockchain network. Smart contracts offer no guarantees if a single node can influence the others or if some nodes have a strong motivation to collude. Thus, decision-makers should analyze the relationships among the nodes before joining a blockchain network. More generally, blockchain does not completely solve trust issues, as peers must still trust that the majority is honest.

Q. SMART CONTRACTS BELONG TO THE SYSTEM

Ownership of smart contracts is a controversial topic. The owner of a smart contract is often its creator. Alternatively, the smart contract can grant special privileges to a specific entity. In any case, however, the smart contract is managed and executed by the nodes of the blockchain system. If a majority of them decide to alter the smart contract, the owner would have no power (or right) to stop them. Thus, the actual owner of a smart contract is the blockchain system itself. While this is a philosophical remark with probably not many practical implications, it is important to underline that the concept of ownership assumes a different meaning in a blockchain system.

IV. CONCLUSION

Smart contracts could revolutionize the world as they are flexible and secure. Nonetheless, smart contracts should not be used as a standalone technology, as they need to be integrated into broad-scope frameworks to express their full potential. To this extent, the standardization of smart contracts is a necessary step.

This study analyzed smart contracts from multiple perspectives and in the context of blockchain technology. In particular, this study: provided multiple definitions for smart contracts; highlighted their main properties and requirements; identified and corrected some common misconceptions on the topic; provided some guidelines for the correct implementation, deployment, and contextualization of smart contracts in a generic standard. This study is the first step toward creating a clear and unified vision of smart contracts. In particular, this study may be insightful for managers, lawyers, and non-technical readers that need to decide on the possible adoption of smart contracts in their fields of expertise.

The analysis developed in this study identified many misconceptions regarding smart contracts, which could hinder the creation of a universal standard. The smart contract name itself is misleading and should be replaced by the more appropriate word chaincode [74]. Sharing a common definition of smart contracts and related properties is a necessity. In particular, a clear understanding of computer science-related topics is fundamental to see through the confusion created by abstractions and misinterpretations.

This study did not analyze smart contracts as a standalone technology, as there are strong connections with the underlying blockchain technology. Consequently, it is necessary to create standards for the proper use of smart contracts and their integration into external technologies and frameworks, which is particularly true for the legitimization and legal recognition of smart contracts. In particular, investigating the following research questions is recommended.

- Which conditions make a blockchain system sufficiently decentralized and secure? Who can trust, repudiate, reject, or ignore the data it stores?
- Which nodes should code/test/execute a given smart contract? Which factors and metrics should determine the degree of reliability of the smart contract?
- Which standards related to the data encoding/decoding should smart contracts follow? Where and how are those standards specified?

- How formalizing which output should a smart contract generate for a given input? For example, how can a formal language define a legal contract? Can declarative languages be used?
- How can a smart contract be associated with its conceptual meaning? For example, if a smart contract compares two numbers, is it comparing two temperatures?
- To which extent can smart contracts replace legal contracts?

The main takeaway from the guidelines proposed in this study is that full decentralization is difficult to achieve and imposes many additional challenges. Given the difficulties of creating truly decentralized smart contracts, we wonder to which extent compromises can be accepted. What is the point of pursuing decentralization if we are ready to sacrifice it for practicality? In the end, are compromise-oriented decentralized solutions more reliable than centralized ones?

REFERENCES

[1] I. Pavlova, "Blockchain ETFs: Dynamic correlations and hedging capabilities," *Managerial Finance*, vol. 47, no. 5, pp. 687–702, Apr. 2021.

[2] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, "Blockchain and smart contracts for insurance: Is the technology mature enough?" *Future Internet*, vol. 10, no. 2, p. 20, Feb. 2018.

[3] H. Hellani, L. Sliman, A. E. Samhat, and E. Exposito, "Overview on the blockchain-based supply chain systematics and their scalability tools," *Emerg. Sci. J.*, vol. 4, pp. 45–69, Aug. 2021.

[4] S. Pan, W. Zhou, S. Piramuthu, V. Giannikas, and C. Chen, "Smart city for sustainable urban freight logistics," *Int. J. Prod. Res.*, vol. 59, no. 7, pp. 2079–2089, Apr. 2021.

[5] A. Ruffini, A. Salerno, and F. Simões, "Net-zero emissions: Main technological, geopolitical, and economic consequences of the new energy scenario," *SSRN Electron. J.*, vol. 2022, pp. 1–14, Apr. 2022.

[6] A. A. Khan, A. A. Laghari, D.-S. Liu, A. A. Shaikh, D.-D. Ma, C.-Y. Wang, and A. A. Wagan, "EPS-ledger: Blockchain hyperledger sawtooth-enabled distributed power systems chain of operation and control node privacy and security," *Electronics*, vol. 10, no. 19, p. 2395, Sep. 2021.

[7] M. Nadini, L. Alessandretti, F. Di Giacinto, M. Martino, L. M. Aiello, and A. Baronchelli, "Mapping the NFT revolution: Market trends, trade networks, and visual features," *Sci. Rep.*, vol. 11, no. 1, pp. 1–11, Dec. 2021.

[8] G. Attanasio, L. Cagliero, P. Garza, and E. Baralis, "Quantitative cryptocurrency trading: Exploring the use of machine learning techniques," in *Proc. 5th Workshop Data Sci. Macro-Modeling Financial Econ. Datasets (DSMM)*, 2019, pp. 1–6.

[9] C. Kohler. (2022). *What is the Blockstream Mining Note?*. [Online]. Available: https://thebitcoinmanual.com/articles/what-blockstream-mining-note/

[10] M. Dell'Erba, "Demystifying technology. Do smart contracts require a new legal framework? Regulatory fragmentation, self-regulation, public regulation," *SSRN Electron. J.*, 2018. [Online]. Available: https://ssrn.com/abstract=3228445, doi: 10.2139/ssrn.3228445.

[11] R. Aringhieri, S. Bigharaz, D. Duma, and A. Guastalla, "Fairness in ambulance routing for post disaster management," *Central Eur. J. Oper. Res.*, vol. 30, no. 1, pp. 189–211, Mar. 2022.

[12] W. Serrano, "Verification and validation for data marketplaces via a blockchain and smart contracts," *Blockchain, Res. Appl.*, vol. 2022, Jul. 2022, Art. no. 100100.

[13] M. Sookhak, M. R. Jabbarpour, N. S. Safa, and F. R. Yu, "Blockchain and smart contract for access control in healthcare: A survey, issues and challenges, and open issues," *J. Netw. Comput. Appl.*, vol. 178, Mar. 2021, Art. no. 102950.

[14] M. Giancaspro, "Is a 'smart contract' really a smart idea? Insights from a legal perspective," *Comput. Law Secur. Rev.*, vol. 33, no. 6, pp. 825–835, Dec. 2017.

[15] A. Janssen and F. Patti, "Demystifying smart contracts," *Osservatorio Diritto Civile Commerciale*, vol. 9, no. 1, pp. 31–50, 2020.

[16] M. Boccia, A. Mancuso, A. Masone, and C. Sterle, "A feature based solution approach for the flying sidekick traveling salesman problem," in *Proc. Int. Conf. Math. Optim. Theory Oper. Res.*, in Communications in Computer and Information Science, vol. 1476, 2021, pp. 131–146.

[17] M. Boccia, A. Mancuso, A. Masone, A. Sforza, and C. Sterle, "A two-echelon truck-and-drone distribution system: Formulation and heuristic approach," in *Optimization and Decision Science*, R. Cerulli, M. Dell'Amico, F. Guerriero, D. Pacciarelli, and A. Sforza, Eds. Cham, Switzerland: Springer, 2021, pp. 153–163, doi: 10.1007/978-3-030-86841-3_13.

[18] A. Diglio, A. Mancuso, A. Masone, C. Piccolo, and C. Sterle, "A MILP formulation for the reorganization of the blood supply chain in italian regions," in *Optimization and Data Science: Trends and Applications*, A. Masone, V. D. Sasso, and V. Morandi, Eds. Cham, Switzerland: Springer, 2021, pp. 51–66.

[19] M. Gajda, A. Trivella, R. Mansini, and D. Pisinger, "An optimization approach for a complex real-life container loading problem," *Omega*, vol. 107, Feb. 2022, Art. no. 102559.

[20] G. Chiaselotti, T. Gentile, and F. Infusino, "Lattice representation with algebraic granular computing methods," *Electron. J. Combinatorics*, vol. 27, no. 1, pp. 1–34, 2020.

[21] *ETSI GS PDL 011 V1.1.1*, Permissioned Distributed Ledger ETSI Industry Specification Group, ETSI, Sophia Antipolis, France, 2021.

[22] G. Perboli, M. Stefano, and R. Mariangela, "Blockchain in logistics and supply chain: A lean approach for designing real-world use cases," *IEEE Access*, vol. 6, pp. 62018–62028, 2018.

[23] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[24] M. Hribernik, K. Zero, S. Kummer, and D. M. Herold, "City logistics: Towards a blockchain decision framework for collaborative parcel deliveries in micro-hubs," *Transp. Res. Interdiscipl. Perspect.*, vol. 8, Nov. 2020, Art. no. 100274.

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decentralized Bus. Rev., 2008. [Online]. Available: https://www.debr.io/article/21260-bitcoin-a-peer-to-peer-electronic-cash-system

[26] N. Szabo, "Formalizing and securing relationships on public networks," *1st Monday*, vol. 2, no. 9, pp. 1–21, Sep. 1997.

[27] V. Buterin, "A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[28] W. Zou, D. Lo, P. S. Kochhar, X.-B.-D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2084–2106, Oct. 2021.

[29] E. Mik, "Smart contracts: Terminology, technical limitations and real world complexity," *Law, Innov. Technol.*, vol. 9, no. 2, pp. 269–300, Jul. 2017.

[30] P. Tolmach, Y. Li, S.-W. Lin, Y. Liu, and Z. Li, "A survey of smart contract formal specification and verification," *ACM Comput. Surv.*, vol. 54, no. 7, pp. 1–38, Sep. 2022.

[31] P. S. Bayón, "Key legal issues surrounding smart contract applications," *KLRI J. Law Legislation*, vol. 9, no. 1, pp. 63–91, 2019.

[32] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: Essential requirements and design options," 2016, *arXiv:1612.04496*.

[33] A. Dixit, V. Deval, V. Dwivedi, A. Norta, and D. Draheim, "Towards user-centered and legally relevant smart-contract development: A systematic literature review," *J. Ind. Inf. Integr.*, vol. 26, Mar. 2022, Art. no. 100314.

[34] S. A. McKinney, R. Landy, and R. Wilka, "Smart contracts, blockchain, and the next frontier of transactional law," *Washington J. Law, Technol. Arts*, vol. 13, p. 313, Apr. 2017.

[35] N. Fotiou and G. C. Polyzos, "Smart contracts for the Internet of Things: Opportunities and challenges," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2018, pp. 256–260.

[36] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 254–269.

[37] W. Brammertz and A. I. Mendelowitz, "From digital currencies to digital finance: The case for a smart financial contract standard," *J. Risk Finance*, vol. 19, no. 1, pp. 76–92, Jan. 2018.

[38] B. Marino and A. Juels, "Setting standards for altering and undoing smart contracts," in *Proc. Int. Symp. Rules Rule Markup Lang. Semantic Web*, 2016, pp. 151–166.

[39] B. Hu, Z. Zhang, J. Liu, Y. Liu, J. Yin, R. Lu, and X. Lin, "A comprehensive survey on smart contract construction and execution: Paradigms, tools, and systems," *Patterns*, vol. 2, no. 2, Feb. 2021, Art. no. 100179.
[40] Y.-W. Jeng, Y.-C. Hsieh, and J.-L. Wu, "Step-by-step guidelines for making smart contract smarter," in *Proc. IEEE 12th Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2019, pp. 25–32.
[41] K. Hu, J. Zhu, Y. Ding, X. Bai, and J. Huang, "Smart contract engineering," *Electronics*, vol. 9, no. 12, p. 2042, Dec. 2020.
[42] D. Macrinici, C. Cartofeanu, and S. Gao, "Smart contract applications within blockchain technology: A systematic mapping study," *Telematics Inform.*, vol. 35, no. 8, pp. 2337–2354, 2018.
[43] T. M. Hewa, Y. Hu, M. Liyanage, S. S. Kanhare, and M. Ylianttila, "Survey on blockchain-based smart contracts: Technical aspects and future research," *IEEE Access*, vol. 9, pp. 87643–87662, 2021.
[44] P. Chapman, D. Xu, L. Deng, and Y. Xiong, "Deviant: A mutation testing tool for solidity smart contracts," in *Proc. Blockchain*, Jul. 2019, pp. 319–324.
[45] M. Saetran, J. Seo, and S. Park, "Leverage sidechains to reduce the workload of smart contracts through parallelization," *J. Comput. Sci. Eng.*, vol. 15, no. 3, pp. 125–133, Sep. 2021.
[46] N. Vashistha, M. M. Hossain, M. R. Shahriar, F. Farahmandi, F. Rahman, and M. M. Tehranipoor, "EChain: A blockchain-enabled ecosystem for electronic device authenticity verification," *IEEE Trans. Consum. Electron.*, vol. 68, no. 1, pp. 23–37, Feb. 2022.
[47] M. Abdelhamid and G. Hassan, "Blockchain and smart contracts," in *Proc. 8th Int. Conf. Softw. Inf. Eng.*, 2019, pp. 91–95.
[48] J. Lánský, "Bitcoin system," *Acta Inf. Pragensia*, vol. 6, no. 1, pp. 20–31, Jun. 2017.
[49] M. Du, Q. Chen, L. Liu, and X. Ma, "A blockchain-based random number generation algorithm and the application in blockchain games," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 3498–3503.
[50] A. Hassan, M. I. Ali, R. Ahammed, M. M. Khan, N. Alsufyani, and A. Alsufyani, "Secured insurance framework using blockchain and smart contract," *Sci. Program.*, vol. 2021, pp. 1–11, Nov. 2021.
[51] V. Capocasale. (2022). *Smart Contract Equivalence*. [Online]. Available: https://github.com/vittoriocapocasale/SmartContractEquivalence
[52] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor, and X. Xu, "On legal contracts, imperative and declarative smart contracts, and blockchain systems," *Artif. Intell. Law*, vol. 26, no. 4, pp. 377–409, Dec. 2018.
[53] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50759–50779, 2019.
[54] F. Schär, "Decentralized finance: On blockchain-and smart contract-based financial markets," *FRB St. Louis Rev.*, vol. 2021, pp. 1–22, Apr. 2021.
[55] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, "Sawtooth: An introduction," 2018. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf
[56] I. Sergey, V. Nagaraj, J. Johannsen, A. Kumar, A. Trunov, and K. C. G. Hao, "Safer smart contract programming with scilla," *Proc. ACM Program. Lang.*, vol. 3, pp. 1–30, Oct. 2019.
[57] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020.
[58] J. Kongmanee, P. Kijsanayothin, and R. Hewett, "Securing smart contracts in blockchain," in *Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng. Workshop (ASEW)*, Nov. 2019, pp. 69–76.
[59] Y. Hu, T. Lee, D. Chatzopoulos, and P. Hui, "Analyzing smart contract interactions and contract level state consensus," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 12, p. e5228, Jun. 2020.
[60] D. Larimer, J. Lavin, N. Hourt, Q. Ma, and W. Prioriello, "EOS.IO technical white paper v2," 2018. [Online]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md
[61] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent advances in smart contracts: A technical overview and state of the art," *IEEE Access*, vol. 8, pp. 117782–117801, 2020.
[62] P. Sreehari, M. Nandakishore, G. Krishna, J. Jacob, and V. S. Shibu, "Smart will converting the legal testament into a smart contract," in *Proc. Int. Conf. Netw. Adv. Comput. Technol. (NetACT)*, Jul. 2017, pp. 203–207.
[63] Y. Liu and J. Huang, "Legal creation of smart contracts and the legal effects," *J. Phys., Conf. Ser.*, vol. 1345, no. 4, Nov. 2019, Art. no. 042033.
[64] A. Norta, "Designing a smart-contract application layer for transacting decentralized autonomous organizations," in *Proc. Int. Conf. Adv. Comput. Data Sci.*, 2016, pp. 595–604.
[65] A. Janssen and M. Djurovic, "The formation of blockchain-based smart contracts in the light of contract law," *Eur. Rev. Private Law*, vol. 26, no. 6, pp. 753–771, Dec. 2018.
[66] K. Lauslahti, J. Mattila, and T. Seppala, "Smart contracts—How will blockchain technology affect contractual practices?" ETLA, Helsinki, Finland, Tech. Rep. 68, 2017.
[67] G. Perboli, V. Capocasale, and D. Gotta, "Blockchain-based transaction management in smart logistics: A sawtooth framework," in *Proc. COMPSAC*, 2020, pp. 1713–1718.
[68] V. Capocasale, D. Gotta, S. Musso, and G. Perboli, "A blockchain, 5G and IoT-based transaction management system for smart logistics: An hyperledger framework," in *Proc. IEEE 45th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2021, pp. 1285–1290.
[69] M. He, H. Wang, Y. Sun, R. Bie, T. Lan, Q. Song, X. Zeng, M. Pustišek, and Z. Qiu, "T^2L: A traceable and trustable consortium blockchain for logistics," *Digit. Commun. Netw.*, 2022, doi: 10.1016/j.dcan.2022.06.015.
[70] A. Gluchowski. (2019). *Introducing Zksync: The Missing Link to Mass Adoption of Ethereum*. [Online]. Available: https://blog.matter-labs.io/introducing-zk-sync-the-missing-link-to-mass-adoption-of-ethereum-14c9cea83f58
[71] M. Almakhour, L. Sliman, A. E. Samhat, and A. Mellouk, "Verification of smart contracts: A survey," *Pervas. Mobile Comput.*, vol. 67, Sep. 2020, Art. no. 101227.
[72] L. Breidenbach *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021. [Online]. Available: https://research.chain.link/whitepaper-v2.pdf
[73] A. Altarawneh, T. Herschberg, S. Medury, F. Kandah, and A. Skjellum, "Buterin's scalability trilemma viewed through a state-change-based classification for common consensus algorithms," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 727–736.
[74] S. Kim, Y. Son, and Y. Lee, "A study on the security weakness analysis of chaincode on hyperledger fabric and ethereum blockchain framework," *J. Green Eng.*, vol. 10, no. 9, pp. 6349–6367, 2020.

VITTORIO CAPOCASALE (Member, IEEE) received the M.Sc. degree in computer engineering from the Polytechnic University of Turin, in 2019, where he is currently pursuing the Ph.D. degree. His main research interests include related to blockchain and its applications to the industrial world.

GUIDO PERBOLI (Member, IEEE) is a Full Professor of decision making and operations research with Politecnico di Torino, an Associate Member with CIRRELT, Québec, Canada, and a Mentor of Startups with an experience of 15 years in Business Development and Research and Development. He is currently a Shareholder and the Chief Scientific Officer of Arisk S.p.a.—a Fintech Startup operating in the usage of AI for the risk prediction and management and Spinoff of Politecnico di Torino. He has authored more than 100 papers on peer-reviewed international journals and conferences and a coordinator of international and national research projects. He is a member of scientific boards and awards, including the Scientific Board of SOS Log and the main Italian Association of Sustainable Logistics.