

Received 4 June 2023, accepted 21 June 2023, date of publication 30 June 2023, date of current version 6 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3290899

SURVEY

A Mapping Review on Cyber-Physical Smart Contracts: Architectures, Platforms, and Challenges

SOFANA ALFUHAID^{1,2}, DANIEL AMYOT¹, (Senior Member, IEEE),
AMAL AHMED ANDA¹, (Member, IEEE), AND JOHN MYLOPOULOS¹

¹School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada

²Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

Corresponding author: Sofana Alfuhaid (salfu014@uottawa.ca)

This work was supported in part by King Abdulaziz University, Saudi Arabia; and in part by the Ontario Research Fund–Research Excellence (ORF–RE) Project CyPreSS: Software Techniques for the Engineering of Cyber-Physical Systems.


ABSTRACT Smart contracts are software systems that monitor, automate, and control the execution of a process, react to violations, and enforce process terms and conditions. There is tremendous interest in developing smart contract applications in banking, finance, insurance, government, and supply chain markets. Many of these applications operate in a cyber-physical environment and adopt architectures based on Internet-of-Things (IoT) and blockchain technologies to support monitoring and ensure data integrity. To investigate how cyber-physical smart contracts are realized for compliance monitoring, together with associated challenges and research opportunities. A mapping review of the literature that surveys underlying architectures and their evaluation. The publications considered came from four databases (Scopus, Web of Science, IEEE Xplore, and Google Scholar), supplemented by manual snowballing. All publications considered are peer-reviewed, written in English, and published in non-predatory venues. A total of 368 publications were considered, with a final selection of 50 papers (all published between 2018 and 2023) that were analyzed along three dimensions: Cyber-physical architectures, infrastructure failures, and technical challenges. Blockchain technologies are the most commonly used platform for smart contracts as they provide decentralized architectures deploying interesting communication patterns, as well as multiple technologies to simplify communication for producing and consuming events. Moreover, such architectures can lead to many types of infrastructure failures including sensor/actuator attacks, network outages, and hardware/software failures, resulting in five important technical challenge areas related to security, availability, robustness, privacy, and legal aspects. Key insights and directions for future research are also reported. This review will inform readers about how cyber-physical smart contracts are being built and deployed and the challenges that are faced by their builders and users.

INDEX TERMS Smart contracts, cyber-physical systems, blockchain, compliance monitoring, events, platforms, mapping review.

I. INTRODUCTION

Smart contracts are software systems that monitor, automate, and control the execution of a process, often of a business or legal nature [1]. There is tremendous interest in developing smart contract applications in diverse markets, including

banking, finance, insurance, government, agriculture, and supply chains.¹ Such systems require special attention to their security and complexity as they often monitor legal transactions and compose components dynamically [2]. Some smart contracts, such as those monitoring bitcoin transactions, operate fully in a cyber environment. Others, such as

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong .

¹<https://tinyurl.com/b8sk3jnk>

smart contracts that monitor meat sale transactions to ensure that delivery complies with perishable food transportation standards, operate in a cyber-physical environment. Our study focuses on this latter class of smart contracts, referred to herein as *cyber-physical smart contracts* (CPSCs).

CPSCs typically deploy Internet-of-Things technologies to monitor and control the execution of a process, often a business process or a legal contract execution. In addition, CPSCs often adopt Distributed Ledger Technologies (DLTs), including blockchain, to ensure integrity and immutability of their data in an environment that handles high-risk transactions without requiring trusted third-parties. However, CPSCs may also involve trusted parties and centralized databases instead of, or in addition to, DLTs.

As CPSC applications multiply, it is important to understand how they are being built and used for supporting compliance monitoring and control. Through a mapping review, this paper identifies and analyses relevant academic literature that is focused on event-based monitoring, including its architectures, platforms, interactions, infrastructure failures, and technical challenges.

As for any literature review, it is important to determine its need and originality. Several literature reviews involving blockchains and smart contracts exist, but with purposes and research questions different from those explored here. Dhairour and Assar [3] systematically analyzed the literature to identify blockchain-based smart contract applications focusing on languages, consensus, and choice criteria, but without mentioning cyber-physical systems (CPSs). Also, Parjuangan and Suhard [4] conduct a review of smart contract platforms and their characteristics in electronic trading. The study of Leka et al. [5] focuses on recent knowledge about smart contract challenges, specifically security vulnerabilities. In a similar work, Vacca et al. [6] conduct a systematic literature review of the current state of the art of smart contract open challenges and available tools and techniques. Furthermore, some studies, such as those from Hasan et al. [1] and Niya et al. [7], have developed systems that utilize smart contracts as software to monitor contract events in CPSs where physical objects are used to report sensor data and events of the environment that is being monitored. Many of these existing studies mention some relevant architectures, work mechanisms, and challenges. However, none of them provides a comprehensive analysis that summarizes existing studies and classifies existing architectures, platforms, trends, infrastructure failures, and challenges of smart contracts for event-based monitoring.

This mapping review is intended for researchers and practitioners who are interested in advancing the engineering tools and methods for building CPSCs, or want to apply them to their professional environment to realize value-adding product and service opportunities.

This review is structured on the basis of four research questions (detailed in section III) that focus on existing architectures, event-based interactions, infrastructure failures, and challenges. Section II first provides background information

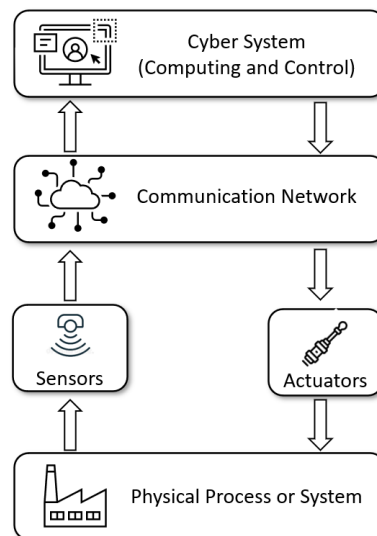


FIGURE 1. Bird's eye view of a typical cyber-physical system architecture.

on CPSCs, while section III covers the methodology used in this mapping review. Then, section IV presents the architectures that have been proposed for CPSCs and shows different patterns and components involved in interacting with their environment to receive and process events. As CPSC architectures may experience infrastructure failures and other challenges, section V highlights such failures and mitigations thereof. Section VI further describes technical challenges while section VII provides explicit answers to the research questions that scoped this study and discusses threats to validity. Finally, section VIII concludes and provides insights into possible future research.

II. BACKGROUND

A. CYBER-PHYSICAL SYSTEMS

The foundational idea behind the concept of CPS is to keep an eye on and exert control over the physical world by incorporating cyber world capabilities (e.g., computation and communication) into aggregations of hardware [8], often composed of IoT devices. Thus, CPSs can be considered as an integration of physical elements, computing systems, and networks within a larger system where they can be controlled and monitored intelligently. Fig. 1 shows a typical CPS architecture in its most abstract terms.

CPS applications include energy systems, smart systems, automotive systems, aerospace systems, robotic systems, industrial systems, IoT applications, and many more. Each of these applications is expected to adapt to changes that come from the outside world and react differently based on the requirements in a safe, secure, efficient, and (ideally) real-time manner [8].

Typically in CPSs, the cyber and physical worlds are exposed to each other through the use of application programming interfaces (API) where the physical devices contain sensors that report actions and states of the environment being monitored [2], [9].

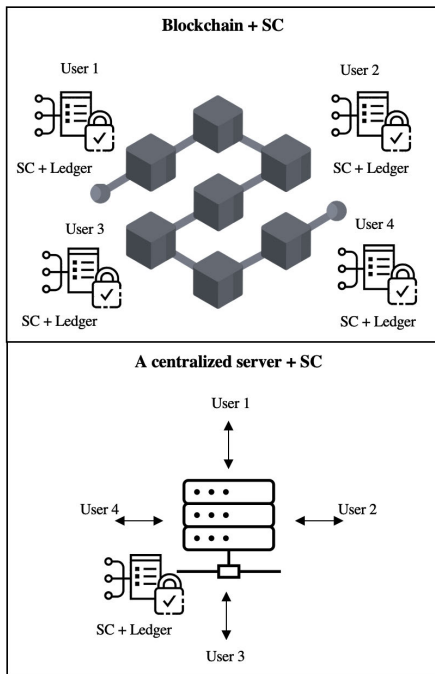


FIGURE 2. Centralized smart contract vs. decentralized smart contract (blockchain).

B. SMART CONTRACTS AND BLOCKCHAINS

The term “smart contract” was first proposed by Szabo in the mid-90’s [10]. It represents the contractual terms of a legal agreement in the real world, but in a completely digital way. These terms are translated and embedded in smart contracts in the form of code that dictates what can and cannot be done, and that is executed automatically based on the terms of the contract. The general goal here is that smart contracts will self-enforce these terms and minimize the need for (trusted) third parties between transnational or transorganizational parties, while obtaining better monitoring and verification where terms must be satisfied [10].

Smart contracts often represent terms or conditions of a legal agreement using functions and events [1], possibly with rule-based patterns to recognize those events [11]. For example, if a contract’s terms specify that *if the shipment exceeds the expected arrival date, then fees must be triggered against the shipping company*, then the arrival of the shipment on a given date must be a recognizable event.

As observed by Niya et al. [7], blockchain is the decentralized ledger technology most commonly used in recent years. Blockchain is a peer-to-peer technology that enables storing and monitoring data in a distributed and decentralized manner [12]. The venue of blockchain platforms has revived the concept of smart contract (Fig. 2) while providing decentralized execution and additional benefits such as immutability and transparency.

Smart contract implementations have capabilities for storing, sending, and receiving data [13]. Implementations can rely on a trusted centralized, decentralized, or some hybrid model [14], [15]. Buterin [16] created a leading DLT

platform, called Ethereum²), that features smart contract capabilities allowing the creation of distributed applications in many areas. Bitcoin,³ which is the first and likely the best-known blockchain platform, supports smart contracts that can process simple transactions. In contrast, Ethereum and other blockchain platforms such as Hyperledger Fabric⁴ can process complex transactions and store records of any data.

There are different types of smart contracts that exist on such decentralized platforms, and they are often developed using different languages:

- 1) Bitcoin-style smart contracts: Use simple instructions as the Bitcoin platform features limited support for conditions, basic arithmetics, logical operations, and cryptography operations (e.g., for verifying digital signatures) on the blockchain [17].
- 2) General-style smart contracts: Use advanced scripts, written in common high-level languages, which are hosted on virtual machines (e.g., deployed using Docker) in order to support the execution of smart contracts on the blockchain. For example, a smart contract for the Hyperledger platform can be written in Java, JavaScript (Node.js), or Go [18].
- 3) Domain-specific smart contracts: Use programming languages that exploit domain-specific knowledge to support contract-related concepts. For example, Ethereum supports Solidity, which features a Turing-complete scripting language for a variety of smart contract applications [19], as well as many other domain-specific programming languages (e.g., LLL, Serpent, and Vyper). Furthermore, many approaches [20], [21], [22] have introduced new specification languages for modeling smart contracts.

There are also three types of blockchain-based ledgers: private, public, and consortium. These differ on the ability of parties to read and write from a ledger, the ability of nodes to join the blockchain network, the ability of nodes to validate and publish a block, and the type of consensus mechanism. For example, only assigned nodes can join the network and validate transactions in a private ledger. However, in a public ledger, anyone can enter the network and publish a new block. Consortium ledgers are in between private and public ledgers [12], [19], [21]. Both Bitcoin and Ethereum are examples of public blockchains, and their respective ledger is available to anyone. Hyperledger Fabric is an example of a private blockchain where the ledger is kept concealed and access is restricted [19].

III. RESEARCH METHODOLOGY

A *mapping review* is designed to provide an overview of the literature relevant to research questions by exploiting academic research databases and complementary search

²<https://ethereum.org/>

³<https://bitcoin.org/>

⁴<https://www.hyperledger.org/use/fabric>

approaches, selecting and analyzing the relevant articles, and synthesizing answers to the research questions. A mapping review also helps identify research gaps and is more oriented towards answering questions than a scoping review, the latter being usually more topic-based and used to scope and characterize the existence of the literature.

To provide explicit and reproducible systematic literature reviews (including mapping reviews), Okoli [23] defined a four-phase methodology that this mapping review follows:

- 1) *Planning*: Planning the review by identifying the review purpose and the research questions.
- 2) *Selection*: Searching and screening the literature for relevant studies.
- 3) *Extraction*: Extracting from the papers the data that is relevant to the research questions, and appraising their quality.
- 4) *Execution*: Answering the research questions by synthesizing answers from the extracted data and reporting the results.

Fig. 3 illustrates the phases involved in the performed mapping review. Also, as done in most literature reviews, a PRISMA diagram [24], [25] is used to summarize the results of various inclusion and exclusion steps of the selection and extraction phases.

A. PLANNING PHASE

This review aims to describe the state-of-the-art for CPSCs. Towards this purpose, the following four research questions have been framed:

- **RQ1**: What are the current platforms that support the implementation of CPSCs for event-based monitoring?
- **RQ2**: How do CPSCs produce and consume events from/to the outside world for monitoring?
- **RQ3**: What are current techniques for mitigating CPSC execution failures in event-based monitoring?
- **RQ4**: What are the main technical challenges faced in the development of CPSCs for event-based monitoring?

B. SELECTION PHASE

To answer the four review questions, an abstract search query that composes the four essential concepts related to this review (smart contract, CPS, monitoring, and event) and their synonyms/variants was designed to find relevant academic papers. The * truncation operator enables matching different variants of a keyword (e.g., for plural forms).

```

`smart contract*`
  AND
(
  architectur* OR cyberphysical
  OR `cyber-physical` OR "cyber physical"
  OR CPS OR platform*
)
  AND
monitor*
  AND
event*

```

TABLE 1. List of databases used in the mapping review.

| Searched Database | Fields |
|-------------------|--|
| Scopus | title/abstract/keywords |
| Web of Science | title/abstract/keywords |
| IEEE Xplore | title/abstract/keywords + full text: event |
| Google Scholar | full text, but simpler query |

TABLE 2. Exclusion and inclusion criteria.

| Exclusion Criteria | Inclusion Criteria |
|---|--|
| 1) Studies that are not in English. | 1) Peer-reviewed scholarly journals and conference papers about event monitoring systems that use smart contracts. |
| 2) Studies in the form of a thesis, book, or survey. | 2) Studies that address at least one of the research questions. |
| 3) Studies that are not peer-reviewed. | 3) Relevant studies obtained through snowballing on previously selected studies. |
| 4) Studies that are duplicates of another study. | |
| 5) Studies whose focus does not answer any of the research questions. | |
| 6) Patents | |

This abstract query was tailored for several databases presented in Table 1. Scopus and Web of Science were included as they are broad-scope, curated databases that cover over 100 million records, and IEEE Xplore (which contains many CPS and smart contract papers) and Google Scholar (again, very broad in scope) were included for their full-text search capabilities.

The concept of “event” in the query was too restrictive when limited to title/abstract/keyword information for Scopus and Web of Science. Therefore, “events” was removed from their query, and a manual full-text search for “events” was conducted. However, as IEEE Xplore supports full-text search, “events” was kept in the full-text search field. For Google Scholar, given the severe limitations of its search engine, a simpler query containing the main keywords was used, and its results were only considered up to a predetermined depth based on Scholar’s ranking of the papers’ relevance.

The review followed an automatic search from the four databases mentioned in Table 1. Additionally, several relevant papers have been found by exploring the referenced works through a complementary backward snowballing strategy (as suggested by Mourão et al. [26]).

The literature search was performed in April 2023. The retrieved papers were first screened using Covidence [27] based on the inclusion and exclusion criteria shown in Table 2. For that part of the screening, title, keywords, abstract, introduction, and conclusion of each papers were reviewed. Articles that met any of the exclusion criteria were excluded and the reasons were noted.

Although there are related patents that exist in that space, e.g., for general contract monitoring [28], smart contract compliance monitoring [29], or the monitoring of smart

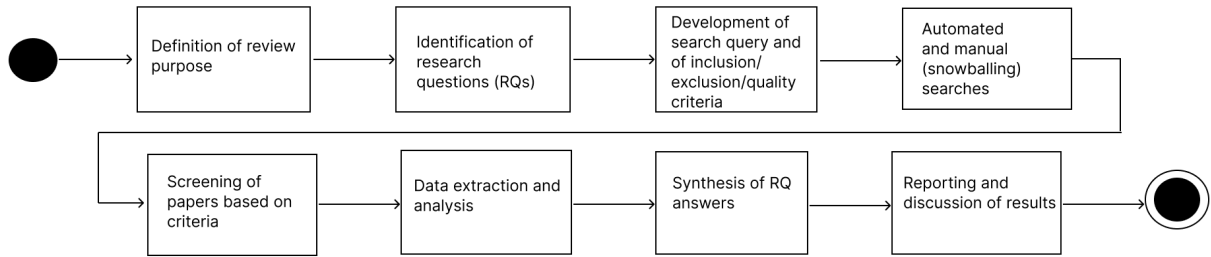


FIGURE 3. Overview of the mapping review methodology.

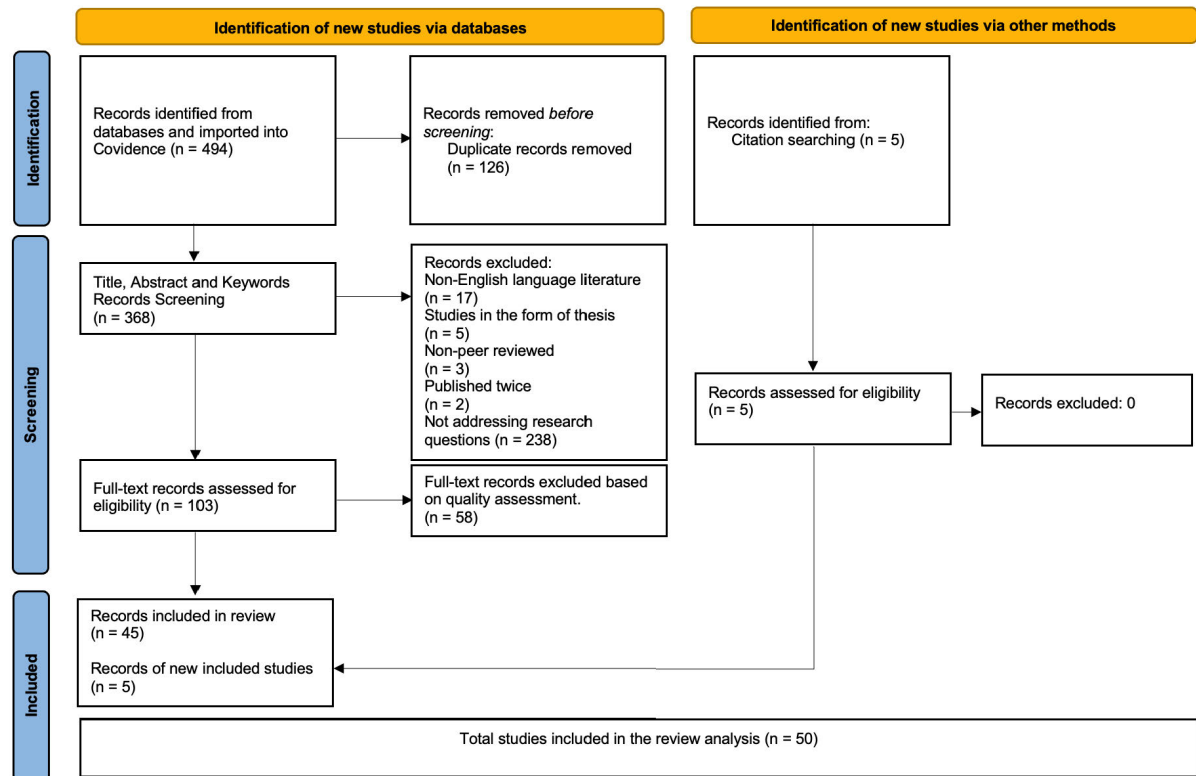


FIGURE 4. Summary of selection results, shown as a PRISMA diagram.

contracts themselves [30], patents were not included as they are not peer-reviewed according to scientific criteria.

C. EXTRACTION PHASE

After completing the first screening iteration mentioned earlier, a second iteration that combined screening and data extraction was conducted, this time using a full-text review. Some articles were excluded based on the quality assessment conducted by the first author. Only articles coming from a non-predatory source of information and meeting at least one of the following assessment criteria were included:

- 1) Clarified the architecture(s) where the smart contract can be used for event-based monitoring.
- 2) Clarified how smart contracts interact, consume, and produce events.
- 3) Discussed infrastructure failures.

- 4) Discussed challenges faced by developers when embedding smart contracts in CPSs.
- 5) Evaluated the architecture(s).

Many studies were excluded because there was neither a clear description of a proposed architecture nor a discussion related to failures or challenges. Some of these papers are however cited in the previous sections and in the discussion as they still provided some useful information to support the content of the review.

A PRISMA diagram [24] summarizing the selection results is presented in Fig. 4.

The analysis was done using Microsoft Excel and is available on Zenodo.⁵ A table was created with different columns used to extract relevant data, including the article

⁵<https://zenodo.org/record/8000387>

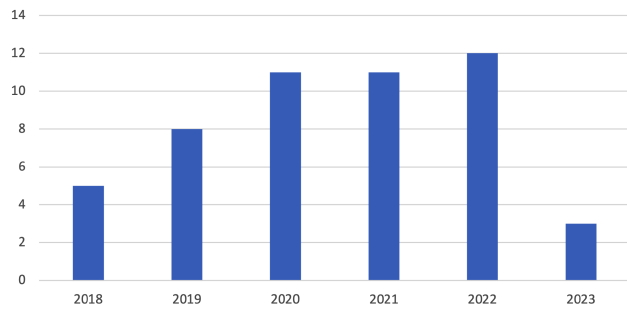


FIGURE 5. CPSC research distribution per year.

information (title, authors, year); the location of the smart contract, its data, and its events in the architecture (on-chain, off-chain, hybrid); the platform used; the deployment technology; the smart contract languages used; the overall approach to event production and consumption; the types of infrastructure observed; and the types of technical challenges faced.

D. EXECUTION PHASE

Answers to the research questions were synthesized by manually clustering the extracted data into different categories related to architecture, failures, and challenges, which are presented in the next sections.

The results overview, presented in Fig. 5, shows that the research on CPSCs started in 2018 with its first publications, and that the highest number of selected publications was in 2022. Note that the queries were run in April 2023, so the results from 2023 are partial.

IV. ARCHITECTURE

In this section, the first and second research questions are answered. The answer to RQ1 on platforms is spread over sections IV-A and IV-C while the answer to RQ2 on events is provided in section IV-B. The first question demands a detailed description of existing architectures for CPSCs, while the second question demands details on how data and events from the environment are produced and consumed while monitoring for compliance.

A. OVERVIEW

Typically, CPSCs consist of several physical (IoT) components that are responsible for collecting data from the outside world and controlling the environment. IoT devices are usually deployed as IoT components that include multiple devices and come in some sort of architecture. These components are networked and controlled by cyber components (e.g., by smart contracts) [2]. Fig. 6 illustrates such physical components, cyber components, and the network. The physical components are usually sensors and actuators [9]. The sensors collect and transmit sensor data through a wired/wireless network to the cyber components. Such an architecture enables compliance monitoring and helps check

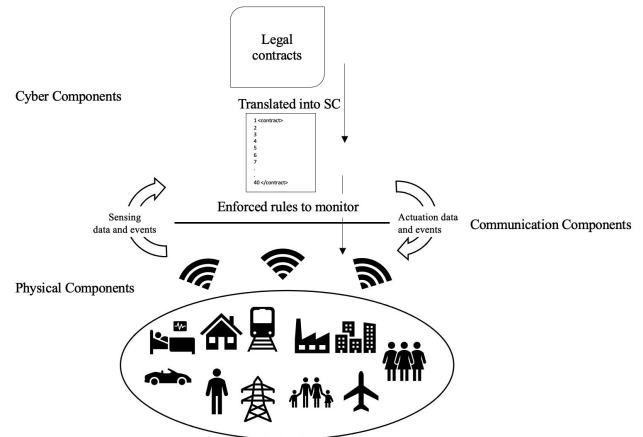


FIGURE 6. Overview of a typical CPSC architecture.

the enforcement of predefined terms of the contract impacting the physical world [1].

Implementations of CPSCs that have been published in the literature are shown in Table 3. They are presented along different architectural characteristics, including smart contract location and data location (on-chain, off-chain, hybrid), platforms, deployment technologies, implementation languages for the smart contract, and the methods used for producing and consuming events. These implementations utilize blockchains as their underlying back-end infrastructure, with the system's operations and business logic encoded in smart contracts. When events or sensor data from the physical world are reported, the smart contract is invoked, and its predetermined terms and rules are executed automatically [9]. In recent years, relatively few studies have focused on the development of smart contracts for centralized systems, compared to decentralized ones [2], as using decentralized blockchain-based architectures offers desirable immutability and transparency features [9].

Blockchain is also employed as a back-end storage solution for storing sensor data and events or for keeping pointers to data stored in another layer of the architecture (off-chain). Storing information on a blockchain is however costly in terms of time, space, energy, and money. Accordingly, some studies have utilized off-chain storage options such as the Interplanetary File System (IPFS) [31], [32], [33], [34], Swarm [32], or Couch DB [11], [35], [36]. These off-chain storage systems allow for more cost-effective and faster alternatives.

Smart contracts are created using either specialized languages like Solidity or general-purpose programming languages like Java, Go, or JavaScript. In the conducted review, Solidity was found to be the most frequently-cited language among the selected papers, as shown in Fig. 7.

Additionally, smart contracts can be deployed on top of centralized (off-chain) or decentralized (on-chain) platforms [2]. However, some recent studies have demonstrated that smart contracts can also be implemented on both

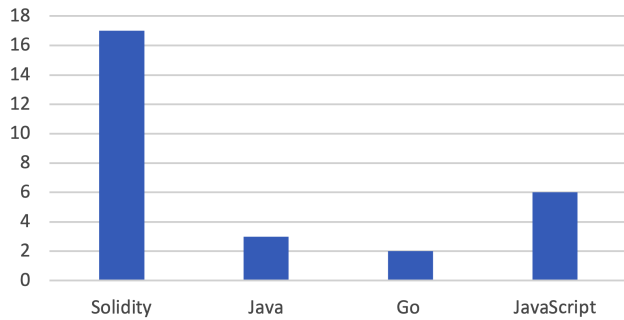


FIGURE 7. Languages used to implement smart contracts in the selected studies.

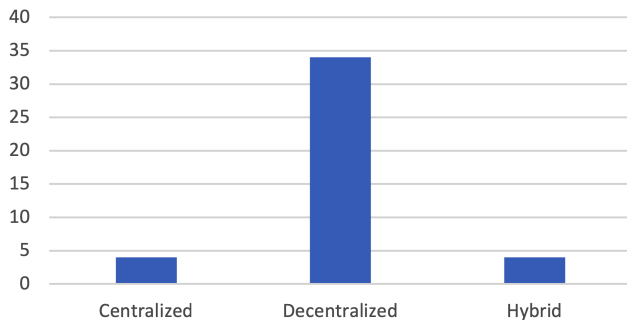


FIGURE 8. Platforms used to implement smart contracts in the selected studies.

centralized trusted third-party and decentralized platforms (hybrid), where a smart contract running on a decentralized platform can trigger another smart contract on a centralized platform, depending on the execution process [14], [15]. Fig. 8 illustrates the current trend of smart contract implementation platforms. Also, among the selected papers, Ethereum and Hyperledger Fabric are the leading blockchain platforms used for the implementation of smart contracts.

Furthermore, generated events play a crucial role in the implementation of smart contracts for monitoring compliance in the physical world. They are used by smart contracts to record or report violations. Event hubs or logs are available in the cyber components of the CPSC, which run on centralized or decentralized platforms and provide access to the events generated by smart contracts to the physical components of the system.

Several studies employ different protocols – such as constrained application protocol (CoAP) and message queuing telemetry transport (MQTT) – to facilitate communication between physical and cyber worlds in regard to sensor data and events [37]. MQTT is the most frequently-cited protocol in the selected studies, which is used for storing, publishing, and aggregating sensor data and events and transmitting them to smart contracts. Section IV-B provides a more detailed description of how smart contracts produce and consume sensor data and events from and to the physical world.

By examining the published literature on the implementation of CPSCs, we have arrived at a partial answer to the first

research question regarding existing architectures for CPSCs, as summarized in Table 3.

B. PATTERNS TO INTERACT WITH THE PHYSICAL WORLD

Having gained insight into the programming languages and platforms involved in constructing CPSCs for compliance monitoring, it is time to address the second research question by providing details on how smart contracts produce and consume events generated from the physical world. This section provides a general overview of the commonly used patterns and components in the literature for maintaining the connection between smart contracts and the physical world.

The ability of smart contracts to enforce contract terms depends on receiving events and data from the physical world. However, on-chain smart contracts cannot interact directly with the physical world; extra components (e.g., data carrier) must be used to maintain the connection between smart contracts and the physical world [42]. This is due to the fact that some DLTs are designed to run smart contracts in isolation to be disconnected from the outside world, offering secure and reliable sharing of contractual agreements of event-driven monitoring [42], [45]. For instance, Ethereum does not allow smart contracts to query data directly from the outside world, but Hyperledger Fabric does [46]. However, a hybrid blockchain has been suggested by Falazi et al. [58] as a way for smart contracts to directly access off-chain data. Regardless, data carriers may still be necessary for any blockchain-based smart contracts to ensure a deterministic behavior of smart contracts in monitoring CPSs [46].

Single-board computers like Raspberry Pi are also used to facilitate communication between the outside world and blockchain-based smart contracts. The Raspberry Pi boards gather sensor data from external sources (e.g., cloud, IoT devices) and execute the relevant function within the smart contract [13], [37]. This causes the smart contract to initiate events to record the new data or contract violations.

Additionally, REST servers offer several REST APIs to connect the outside world to blockchain-based smart contracts [9]. These servers allow web applications or physical components to interact with smart contracts to access monitored data and events. Similarly, Ethereum provides several APIs (known as Web3 APIs) for calling back events monitored by smart contracts from the physical world [13].

Furthermore, an oracle acts as a data carrier or a mediator to establish a secure connection between blockchain-based smart contracts and external components such as APIs, IoT devices, cloud providers, and more [38], [39], [42], [43], [44], [45], [46], [47], [49], [59]. Fig. 9 illustrates the basic architecture of an oracle in conjunction with smart contracts and Table 4 provides a description of the function of each type besides other patterns that enable physical components and smart contracts to communicate with each other in interconnected ways.

Based on the reviewed literature, all of the above-mentioned technologies can be leveraged in centralized, decentralized,

TABLE 3. Relevant literature related to CPSCs that includes a proof of concept.

| Article | Year | Data Location | SC Location | Platform | Deployment | SC Languages | Events | Approach to Event Production and Consumption |
|---------------------------|------|---------------|-------------|--------------------|------------|--------------|-----------|--|
| Niya et al. [7] | 2018 | hybrid | on-chain | Ethereum | - | Solidity | on-chain | - |
| Wright et al. [38] | 2018 | hybrid | on-chain | Ethereum | Truffle | Solidity | on-chain | publish/subscribe |
| Neidhardt et al. [39] | 2018 | off-chain | on-chain | Ethereum | - | Solidity | on-chain | - |
| Zhou et al. [40] | 2018 | off-chain | on-chain | Ethereum | - | Solidity | on-chain | - |
| Smirnov et al. [41] | 2018 | off-chain | on-chain | Hyperledger Fabric | Docker | GO/Java | on-chain | publish/subscribe |
| Hasan et al. [1] | 2019 | off-chain | on-chain | Ethereum | Remix IDE | Solidity | on-chain | message queues (MQTT) |
| Bagozi et al. [31] | 2019 | off-chain | on-chain | Ethereum | - | Solidity | on-chain | - |
| Liu et al. [42] | 2019 | off-chain | on-chain | Ethereum | - | Solidity | on-chain | message queues (MQTT) |
| Rúbio et al. [14] | 2019 | - | hybrid | Ethereum | - | - | - | - |
| Uriarte et al. [43] | 2019 | off-chain | hybrid | Ethereum | - | - | on-chain | - |
| Lopez-Pintado et al. [32] | 2019 | off-chain | on-chain | Ethereum | - | Solidity | on-chain | - |
| Hang & Kim [9] | 2019 | hybrid | on-chain | Hyperledger Fabric | Docker | JavaScript | on-chain | Message Broker |
| Breiki et al. [44] | 2019 | hybrid | on-chain | Ethereum | - | Solidity | - | message queues (MQTT) |
| Lockl et al. [13] | 2020 | hybrid | on-chain | Ethereum | Truffle | Solidity | on-chain | read/write |
| Tahmasebi et al. [33] | 2020 | off-chain | on-chain | Ethereum | Remix IDE | Solidity | off-chain | publish/subscribe |
| Solaiman et al. [15] | 2020 | off-chain | hybrid | Ethereum | Remix IDE | Solidity | on-chain | message queues (MQTT) |
| Ladleif et al. [45] | 2020 | off-chain | on-chain | - | - | - | on-chain | publish/subscribe |
| Dienbauer et al. [46] | 2020 | off-chain | on-chain | Hyperledger Fabric | Docker | JavaScript | on-chain | - |
| Taghavi et al. [47] | 2020 | off-chain | on-chain | Ethereum | Remix IDE | Solidity | on-chain | - |
| Lehnert et al. [48] | 2020 | off-chain | on-chain | Ethereum | Remix IDE | Solidity | - | - |
| Kochovski et al. [49] | 2020 | off-chain | on-chain | Ethereum | - | - | on-chain | - |
| Jamil et al. [50] | 2020 | off-chain | on-chain | Hyperledger Fabric | Docker | - | on-chain | - |
| Hang et al. [37] | 2020 | off-chain | on-chain | Hyperledger Fabric | Docker | JavaScript | on-chain | - |
| Hang & Kim [11] | 2020 | off-chain | on-chain | Hyperledger Fabric | Docker | JavaScript | on-chain | request/response |
| Baralla et al. [51] | 2021 | hybrid | on-chain | Ethereum | - | Solidity | on-chain | read/write |
| Cacciagrano et al. [52] | 2021 | hybrid | hybrid | Ethereum | - | - | on-chain | - |
| Alzubaidi et al. [35] | 2021 | on-chain | on-chain | Hyperledger Fabric | Docker | Java | on-chain | read/write |
| Shukla et al. [53] | 2021 | on-chain | on-chain | Ethereum | - | Solidity | on-chain | read/write |
| Jamil et al. [36] | 2021 | hybrid | on-chain | Hyperledger Fabric | Docker | JavaScript | on-chain | request/response |
| Jamil et al. [54] | 2022 | hybrid | on-chain | Hyperledger Fabric | Docker | JavaScript | on-chain | - |
| Pustišek et al. [55] | 2022 | hybrid | on-chain | Ethereum | - | Solidity | on-chain | publish/subscribe |
| Hewa et al. [56] | 2022 | hybrid | on-chain | Hyperledger Fabric | Docker | Java | on-chain | message queues (MQTT) |
| Zhang et al. [57] | 2022 | hybrid | on-chain | Hyperledger Fabric | Docker | Go | on-chain | - |
| Kumar et al. [34] | 2023 | off-chain | on-chain | Ethereum | - | - | on-chain | - |

or hybrid models for CPSCs, except for the Web3 API, a built-in functionality provided by the Ethereum blockchain that can be utilized on either decentralized or hybrid models.

C. LAYERED ARCHITECTURE OF CPSC

From the literature listed in Table 3 and according to Governatori et al. [61], there are three main approaches for applying smart contracts for compliance monitoring in CPSCs:

- Centralized (off-blockchain implementation)(off-chain): The smart contract is managed by a third-party server, the contract's terms and conditions are monitored/carried out off-chain, and contract-related data and events are stored off-chain.
- Decentralized (blockchain implementation)(on-chain): The smart contract is deployed on the blockchain, the

contract's terms and conditions are monitored/carried out on-chain, and contract-related data and events are stored on-chain.

- Hybrid (off/on-chain implementation): The smart contract can be divided where a part of the smart contract is placed on-chain, while the other part remains off-chain. Some of the contract's terms and conditions are monitored/carried out off-chain, while the rest is monitored on-chain. Some contract-related data and events are stored off-chain, and the rest is stored on-chain.

Based on the reviewed literature, different architecture implementations were chosen by developers based on criteria such as privacy and scalability [15]. However, blockchain-based architectures were the most cited ones among selected papers. Few studies investigate off-chain and hybrid architectures.

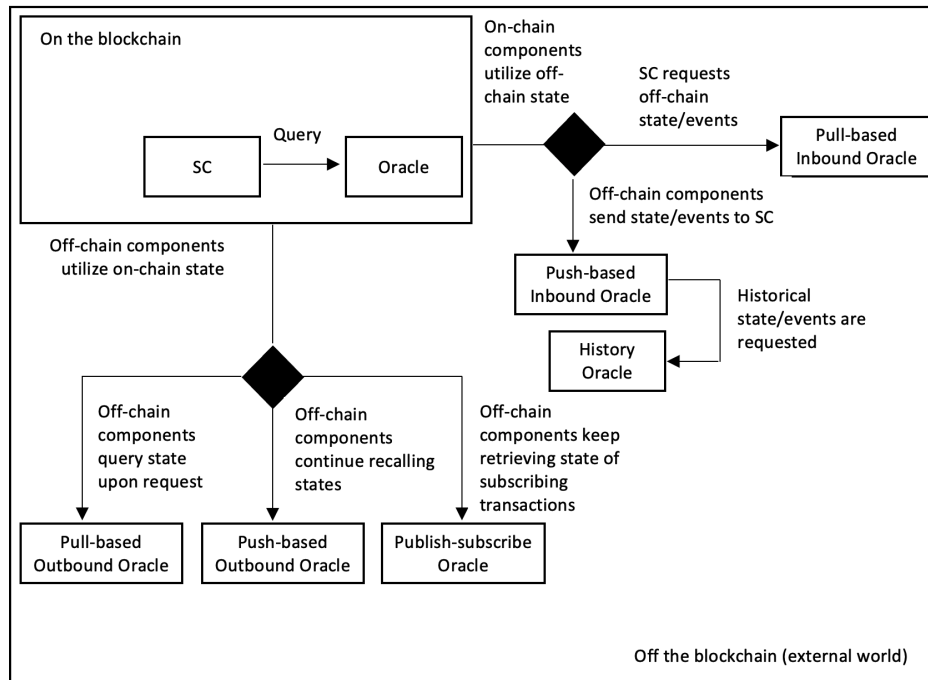


FIGURE 9. An illustration of basic oracle patterns to access data from/to blockchain and smart contracts in CPSCs.

TABLE 4. Patterns used by smart contracts to interact with the physical world for consuming and producing sensor data and events.

| Article | Pattern | Purpose |
|--|--------------------------|---|
| [1, 7, 11, 13, 38, 48, 51, 56] | Raspberry Pi board | Catches off-chain sensor data or events and calling suitable function within SC. |
| [2, 9, 11, 14, 15, 32, 36, 37, 42, 46, 49, 50, 53, 60] | REST APIs | Enables web applications or physical devices to invoke smart contracts to access monitored data and events. |
| [13, 15, 33, 34, 38, 40] | Web3 APIs | Calls back events monitored by smart contracts from the physical world. |
| [39, 42, 43, 46, 47, 49] | Off-chain oracle | Fetches off-chain sensor data and events from multiple data sources in the physical world, verify them and deliver them to SC. |
| [42] | On-chain oracle | Listens to off-chain data requested by SC. |
| [45] | History oracle | Provides history values besides the current or latest value |
| | Publish-subscribe oracle | Retrieves new values immediately from multiple off-chain data sources when changes happen as long as transactions have been subscribed. |
| [59] | Inbound oracle | Sends off-chain data to on-chain components. |
| | Outbound oracle | Allows smart contract to request data from on-chain components. |

Several design concepts have been proposed for using smart contracts to monitor compliance by utilizing sensor

data and events generated by IoT devices. Part of the literature suggests a monitoring architecture that includes a front-end monitoring application to observe sensor data and events produced by the physical world [7], [13], [33]. Typically, external applications initiate smart contracts to monitor and enforce contract terms, while smart contracts store sensor data and record related events. In their architecture, smart contracts are deployed, installed, and instantiated on every node on the blockchain network, which serves as a back-end storage and processing infrastructure.

Hasan et al. [1] proposed a blockchain-based smart contract for monitoring product shipments, defined by business rules and risk thresholds. The physical world’s IoT-enabled containers and sensors are used to track the movement of the product and perform self-checks of various measurements, comparing them to predefined conditions. An MQTT server is employed to store, aggregate, and regularly publish sensor readings. If there is a violation, the container will initiate a call to the smart contract, triggering and registering an event in the events log. A similar approach was proposed by Lockl et al. [13]; however, the main difference is that a monitoring dashboard application is available to end-users, allowing them to monitor sensor data and related events and register components. Additionally, the blockchain is used as a light node, storing only the hash of blocks rather than the entire blocks. In another similar approach proposed by Hang and Kim [9], smart contracts are utilized to store sensor data and keep track of the configuration of physical components. The contracts trigger events when predefined monitoring conditions are met or violations occur. A client application can

send a request via REST APIs to the smart contract to either register a new component for monitoring or access stored sensor data and events. CoAP is employed by the server to transmit sensor data from devices to smart contracts in real-time.

Furthermore, a four-layered architecture for CPSCs was proposed by Tahmasebi et al. [33], consisting of application, service management, gateway, and physical layers. The gateway layer collects and saves the sensor data from the real world in off-chain storage. Smart contracts are triggered by the aggregated data and executed accordingly. An implementation of two smart contracts on the blockchain monitors the execution and registration of IoT devices. The sensor data produced by these devices is stored in off-chain storage using IPFS. Bagozi et al. [31] also presented a CPSC in a four-layered architecture that includes the acquisition, gateway, blockchain, and application layers. However, they use smart contracts to provide anomaly detection services based on the sensor data and related events from monitored devices.

Zhou et al. [40], Lopez-Pintado et al. [32], and Kochovski et al. [49] propose decentralized architectures for CPSCs similar to the architecture of the above-mentioned literature, including physical, gateway, distributed (blockchain), and application layers. It is worth mentioning that some studies have combined some layers or used different terminology. For instance, Lopez-Pintado et al. [32] proposed a three-layered architecture of CPSCs for compliance monitoring consisting of storage, access, and process-aware layers. Zhou et al. [40], Lopez-Pintado et al. [32], and Kochovski et al. [49] differ from the above literature by adding an extra layer for off-chain events monitoring. Zhou et al. [40] introduced a witness model to report violations and monitor off-chain events stored in the cloud. In contrast, Lopez-Pintado et al. [32] added an off-chain event monitoring model to retrieve and monitor on-chain events stored on an event log. On their side, Kochovski et al. [49] added a decision-making layer with a decision-making mechanism, a monitoring system, and an orchestration system for off-chain monitoring.

The literature highlights the importance of various components such as devices, event management, off-chain storage, communication interface, off-chain monitoring systems, and client applications that are connected using decentralized, centralized, or hybrid architectures. Fig. 10 presents a typical tier architecture for compliance monitoring of CPSCs, obtained from the reviewed literature. CPSCs can have multiple independent layers that developers can modify. However, the minimum requirements for building CPSCs are displayed in the figure and described as follows:

- 1) **Physical tier:** This layer contains physical devices (e.g., sensors and actuators), data storage, communication protocols, and so on. Physical devices collect sensor data and events from the physical world and pass them to the next tier.
- 2) **Delivery, Aggregation, and Control tier:** This tier is composed of data carriers that validate and verify the

TABLE 5. Data storage for producing and consuming events.

| Article | Data Storage |
|-----------------------------|-------------------------------|
| [1, 9, 44] | CoAP servers |
| [9, 15, 38, 41, 42, 44, 45] | Message queues (MQTT) servers |
| [33, 56, 62] | Fog node |
| [44] | Cloud storage |

data generated from the physical world before sending it to the service tier. It may also contain monitoring agents that process the data received from sensors for compliance monitoring and trigger the appropriate smart contract function in the service tier. Each of them has communication capabilities to communicate with the service tier.

- 3) **Service tier:** This is the tier where the smart contracts are placed to pull information from the physical world to apply the agreed policy from the agreed legal contract. Also, in this tier, events are generated based on the compliance monitoring rules specified by smart contracts. All events, sensor data, and device information are stored in this tier as well. Fig. 10 shows three main options, where SC execution and monitoring can occur on-chain, off-chain, or both on-chain and off-chain.
- 4) **Application tier:** This tier provides many services, such as monitoring the execution of the physical world (e.g., a monitoring dashboard), allowing users to interact with the data, or performing analysis on them.

Additionally, based on the reviewed literature, there are several options for storing collected sensor data and events generated by the physical world, where smart contracts can access data/events through the control layer for compliance monitoring purposes. Table 5 lists various data storage options where sensor data and events are being consumed from and produced for the physical world.

V. INFRASTRUCTURE FAILURES

After reviewing existing architectures for CPSCs, and the way events are generated and consumed in the environment that is being monitored, we focus here on RQ3. Specifically, we study possible infrastructure failures and summarize existing approaches to alleviating these failures. Fig. 11 shows the risks that are identified from the reviewed literature and the corresponding mitigation approaches.

The first noticeable risk is that sensor data and related events that are generated by the physical world are not sent directly to the smart contracts; rather, they are stored and transmitted through a third party (i.e., data carrier) that resides on a centralized architecture. This approach is particularly vulnerable because it consists of a single point of failure [47]. Taghavi et al. [47] suggest utilizing multiple data carriers to feed the data to smart contracts from the outside world as a mitigation approach.

Another risk is that CPSCs rely on centralized services (e.g., cloud, fog node, MQTT, and CoAP servers) for storing

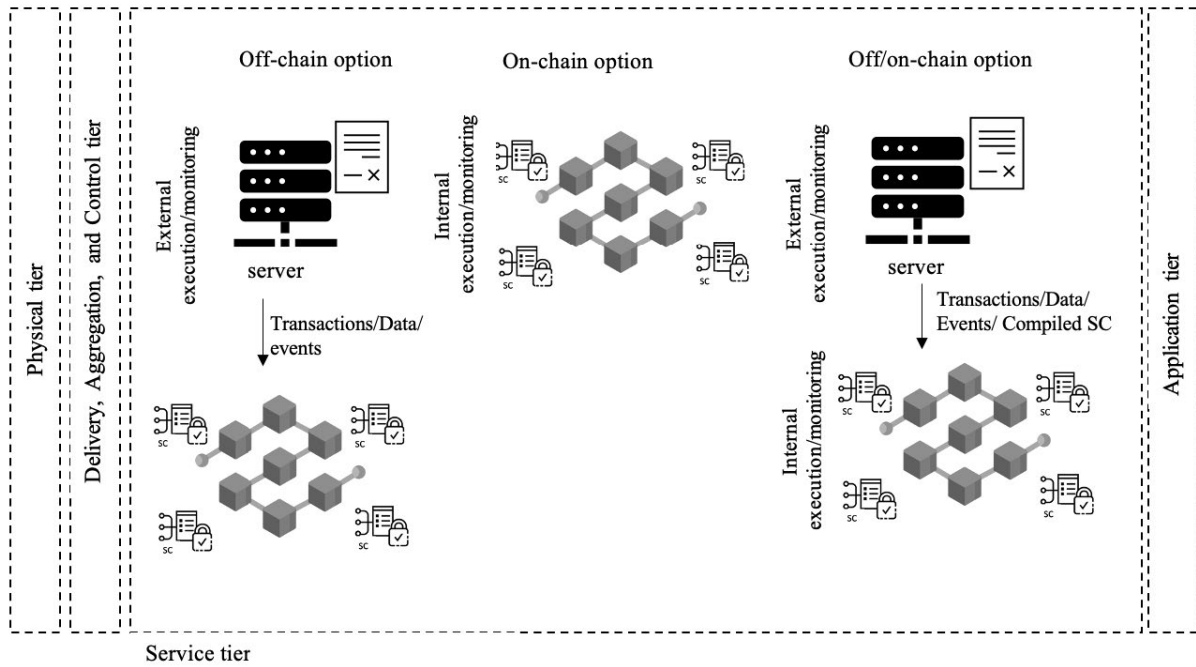


FIGURE 10. Conceptual tier architecture for CPSCs.

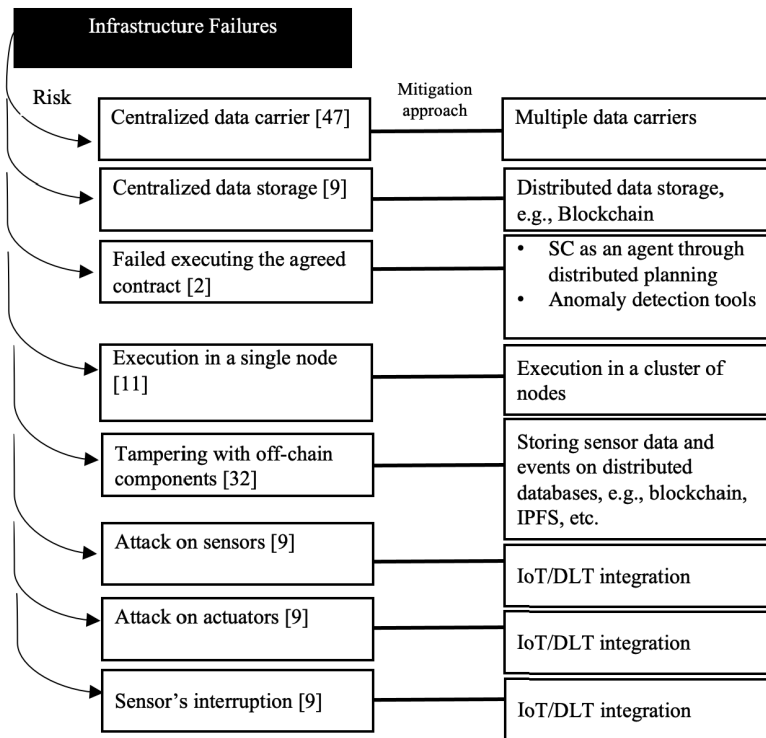


FIGURE 11. Common CPSC infrastructure failures with mitigation approaches.

physical device information and performing operations on them. However, a disadvantage here is, again, a single point of failure for such services [9]. The suggested mitigation approach is to combine centralized storage with distributed

storage, such as blockchains [9]. This concern is exemplified by the Atlanta Ransomware Attack⁶ in 2018, where a

⁶<https://18.nu/qh9M>

ransomware incident targeted the centralized infrastructure of the city, causing disruptions to critical services. This incident serves as a stark reminder of the risks associated with relying solely on centralized services in CPSCs. To enhance resilience and mitigate the impacts of such infrastructure failures, it becomes crucial to explore decentralized and distributed storage solutions, such as blockchains, to ensure the availability and integrity of CPSC operations [9].

Additionally, monitoring processes that involve multiple agents entails the risk of failures resulting from agents not following the agreed protocol of execution. Shukla et al. [2] propose CPSCs that monitor process execution and detect anomalous executions (centralized or decentralized). For instance, the Binance Smart Chain exploit in 2022 serves as a relevant example.⁷ In this incident, an attacker exploited a vulnerability in the Binance Smart Chain's infrastructure, manipulating a smart contract's code to steal a significant amount of "FOO" tokens. Such incidents emphasize the importance of implementing robust security measures and conducting careful code reviews to prevent unauthorized access and manipulations of smart contracts.

Moreover, running a CPSC on a single node can result in system failure if the node fails. Hang and Kim [11] utilize blockchain as back-end storage for compliance monitoring. However, their system runs on a single node, which makes it fault-intolerant. The authors do point out that the usage of a single (solo) node is suitable for running smart contracts for testing purposes only. However, for production purposes, they suggest using clustering nodes for implementation, such as those used by Kafka⁸ or Raft⁹ to avoid system failures.

Furthermore, components that reside off-chain are vulnerable to tampering. In this context, Lopez-Pintado et al. [32] suggest the use of distributed data storage, such as blockchain, to store sensor data and related events. This mitigation approach provides reliable and secure storage, so all compliance monitoring processes happen on-chain.

Typically in CPSCs, IoT contains multiple sensors that are responsible for producing and exchanging massive quantities of data. Such sensors are themselves vulnerable to cyberattack and hence compromise the integrity of the data. IoT/DLT integration is suggested as a paradigm to handle such attacks [9].

Intercepting and altering the order of received data is another risk that can cause major losses. IoT/DLT integration can provide a level of authenticity of IoT devices, such as actuators, to ensure the integrity of transmitted data [9].

Finally, good network connectivity is crucial for IoT devices to perform properly and transmit sensor data to the smart contract; failure to do so can result in significant losses of sensor data and events [9].

⁷<https://18.nu/qhbw>

⁸<https://kafka.apache.org/>

⁹<https://raft.github.io/>

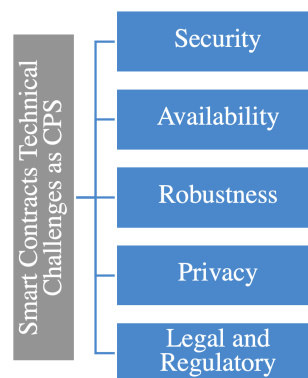


FIGURE 12. Technical challenges for implementing CPSCs.

VI. TECHNICAL CHALLENGES

On the basis of our mapping review, five main challenges were identified as an answer to RQ4, as shown in Fig. VI: Security, availability, robustness, privacy, and legal and regulatory aspects.

A. SECURITY

Smart contracts cannot directly access off-chain data accumulated by monitoring the outside world. Instead, access needs to be provided by a third-party data carrier [38], [39], [42], [43], [45], [46], [47], [49], [59]. This is because smart contracts have been designed to operate in a closed environment, disconnected from the outside world for security reasons [19], [42], [45]. Oracles have been used as a data carrier solution to fetch data from different data sources, verify them, and send them to the smart contract [42], [44], [45], [46], [47]. However, the trustworthiness of oracles and the integrity of provided data have become challenging. Therefore, the lack of trusted data carriers and the absence of a robust environment of reliable data sources impede the applicability of CPSCs.

Another concern is the lack of security measures for the communication between physical and cyber components used to link smart contracts with the outside world [63]. For example, the communication between REST APIs and physical components (e.g., IoT) [11], [37], [50]. Similarly, as more heterogeneous physical and cyber components and services become connected to cyber-physical smart contract systems, more security risks, insecure connections, and bugs are expected to happen [64].

To deal with those smart contract vulnerabilities and take advantage of the CPSC paradigm, blockchain-based smart contracts have been proposed as a solution because they provide distributed security with their cryptographic mechanisms, especially when sensor data and events are collected and shared among different physical and cyber components [64]. Additionally, security analysis tools for contract vulnerabilities are constantly being developed to find potential security bugs and check compliance and potential violation of contract behavior.

Two relevant open source audit tools are Securify¹⁰ and Manticore.¹¹ Securify is a security audit tool for Ethereum smart contracts. It uses symbolic analysis on the dependency graph of the contract to obtain accurate semantic details from the code, and then it checks whether the smart contract behaves according to what is intended. Manticore is another symbolic execution tool capable of tracing smart contract transaction inputs to detect potential violations.

Furthermore, an Intrusion Detection System has been developed to handle other vulnerabilities and cyber and physical attacks as a result of the increasing incorporation of cyber and physical components where sensor data and events are collected and used to monitor different features of a CPSC [31], [34], [62], [63], [65]. An Intrusion Detection System is a software tool that can be used to detect attacks automatically using machine learning, deep learning, or other techniques [34]. It is responsible for detecting attacks, triggering warnings, or taking action [34]. For example, Bagozi et al. [31] developed an anomaly detection service to monitor sensors and check if identified measures of sensor data are above or about to reach pre-identified thresholds during a transportation journey. Kumar et al. [34] propose a scalable blockchain-based smart contract for secure data transmission. Also, A deep learning architecture combining Deep Sparse AutoEncoder with Bidirectional Long Short-Term Memory is proposed for intrusion detection in a healthcare network. Kumar et al. [63] discuss IoT-based zero-touch networks for secure data sharing. They also suggest a deep learning approach for intrusion detection. Kumar et al. [62] suggest a two-level approach for data security. The first level uses blockchain and smart contracts to ensure secure data exchange, while the second level utilizes deep learning to encode data into a new format that prevents attacks. Kumar et al. [65] suggest a new framework for secure data sharing in Intelligent Agriculture that utilizes deep learning and smart contracts enabled by the Internet of Things to detect intrusions.

Note that the above technological solutions to security concerns are those that focus on cyber-physical smart contracts, as scoped by our research questions and search query. Other solutions to security vulnerabilities targeting other types of systems can be found in the work of Leka et al. [5].

B. AVAILABILITY

Availability in cyber-physical smart contracts relates to the capacity of smart contracts to maintain their state and be accessible to entities and physical and cyber components, regardless of the circumstances, e.g., power outages, network outages, attacks, or resource limitations [13]. For example, using Wireless Sensor Networks as a network infrastructure constitutes a risk because of limitations with respect to connectivity, data stream, energy, storage, and capacity [33]. Also, the current CPSC paradigm relies on transmitting

sensor data and events to off-chain centralized servers that represent single points of failures before delivering them to smart contracts for compliance monitoring [13].

Maintaining CPSC availability is challenging, especially in scenarios where it is critical to ensure data is always available. For instance, such scenarios include monitoring sensor data and events in healthcare systems or food supply chains, which can be addressed by ensuring continuous functioning of both cyber and physical components even if some of them are damaged, and by implementing solutions to handle potential single points of failure [13].

To tackle the challenges related to availability, the CPSC paradigm can integrate various recommended methods, such as redundancy and load-balancing capabilities. An IoT/DLT architecture has often been used as a back-end infrastructure as it maintains redundancy [13]. Blockchain-based architectures allow cyber and physical components to be distributed over multiple nodes handling the same tasks; there is no single point of failure. Thus, when one node fails, the other nodes can take over [37]. Another approach is to use load-balancing, which allows deploying physical and cyber components over different resources to avoid overloads [49].

C. ROBUSTNESS

Smart contract robustness is crucial for maintaining contact with the physical world via actuators and sensors. Currently, there is a significant rise in the adaptation of physical components such as IoTs due to their capabilities to provide real-time data and enable networking among various CPS applications [66]. Also, the amount of generated real-time data by IoTs is increasing, which requires massive storage, processing techniques, and networks allowing them to interact with other physical and cyber components and exchange data for compliance monitoring conducted by smart contracts [66]. Thus, since CPSCs involve growing cyber and physical components and data, it is hard to maintain robustness and expect how the systems can react to different conditions, e.g., power constraints, network outages, IoT-limited resources, and so on. Also, such systems adopt a centralized architecture that is exposed to data loss at any given time, as such architecture is not fault-tolerant [66].

To address these challenges, many studies, such as [13] and [66], report on architectures that achieve fault-tolerance through redundancy, which ensures that if one component stops responding for any reason, other components can carry out. Another robust mechanism is the use of artificial intelligence and expert systems to make decisions about an CPSC's behavior and take actions accordingly, e.g., redistribute resources to avoid overwhelming IoT devices of the monitored environment [66].

D. PRIVACY

As indicated earlier, many CPSCs studied in the literature adopt an IoT/DLT architecture. However, maintaining sensory data on-chain is expensive. Therefore, several studies

¹⁰<https://github.com/eth-sri/securify2>

¹¹<https://github.com/trailofbits/manticore>

have turned to alternate hybrid approaches for storing sensor data and physical device information. Such practices are prone to privacy violations, particularly in terms of data privacy for smart contracts and the privacy of data carriers that contain sensitive data concerning the monitored process [13], [51].

Currently, public blockchains represent a real challenge as they do not support access control, resulting in unrestricted access to contract-related data and sensor data, as access to sensitive data can not be prevented. Thus, a variety of solutions have been suggested, such as (1) zero-knowledge proofs [34], [56], [67], (2) homomorphic encryption [57], and (3) secure multi-party computation [57]. Kosba et al. [67] proposed a model called Hawk that allows developers to write a privacy-preserving smart contract using zero-knowledge proofs that can keep its privacy to prove the validity of transactions to contract parties without exposing its content. Also, Kumar et al. [34] suggested a privacy-preserving scheme to protect and prevent data leakage during the transmission of data from/to smart contracts. This privacy-preserving scheme involves verifying IoTs using zero-knowledge proofs. Hewa et al. [56] utilized zero-knowledge proof to maintain anonymity of identities stored in blockchains. Other privacy-preserving primitives suggested by Zhang et al. [57] include homomorphic encryption and secure multi-party computation. Homomorphic encryption keeps data confidentiality by conducting operations on encrypted data [38], whereas secure multi-party computation is another type of encryption that allows parties to come together to conduct computation on off-chain functions without disclosing their data [15].

For Data carriers, they do not all have the same level of privacy and data access because sensor data and events are collected and shared among different off-chain data sources in CPSs. Therefore, the privacy and integrity goals of data carriers are challenging. Town Crier [68] is an approach that provides controlled access to off-chain data provided by different data carriers.

It is worth noting that these privacy-preserving approaches are resource intensive in a cyber-physical smart contract environment full of IoT devices with limited capacity.

E. LEGAL AND REGULATORY

As stated earlier in section II, a smart contract is a program created using a programming language, such as Go, JavaScript, or Solidity, that can independently enforce, verify and control the execution of a legal contract. However, a smart contract may not always be considered a completely enforceable legal contract, depending on how well it satisfies the requirements of laws and legal standards [69]. Thus, there could be an inconsistency between the legal contract and its corresponding digital representation, which may affect the codification of laws [69]. Recent research initiatives related to this matter [20], [21], [22] aim to enable the development and verification of legal contracts and their corresponding smart

contracts through the use of domain-specific languages with various levels of formality.

There could also be legal issues with the data manipulated by smart contracts. For example, in European countries and in the US, people's medical data are protected under privacy protection regulations and governance [50], which allows individuals to ask for their personal information to be removed; this is the so-called "right to be forgotten". Such provision conflicts directly with the immutability feature of smart contracts that leverage blockchain technology.

In addition, the execution of smart contracts is a dynamic process that cannot occur in isolation, as it is often influenced by various factors and external forces [61]. These factors involve rules and regulations (the law in a particular jurisdiction) from other contracts and external events from IoT devices that may affect the contract's outcome.

Thus, legal and regulatory challenges include, but are not limited to:

- Determining which legal rules and regulations would apply to transactions being executed in CPSC applications;
- Deciding on a strategy for the modification and deletion of data from the blockchain (e.g., through access removal or blockchain forking), as required by applicable legislation;
- Determining who takes responsibility for the consequences (disputes, claims, and financial penalties) when the contract's outcome does not align with the legal requirements that must be met.

Collaboration among legal experts, engineers, and stakeholders is essential to tackle the legal and regulatory aspects of smart contracts. Such collaborative effort can help ensure the integration of legal requirements into technical design and implementation, as well as help guarantee code correctness, compliance, and regulatory alignment during the deployment and execution of smart contracts. Approaches such as formal verification and others surveyed by Wang et al. [19] can help smart contracts become better aligned with existing jurisdictions.

VII. DISCUSSION

This section explicitly answers the four research questions identified in this mapping review, together with relevant threats to the validity of our work.

A. ANSWERS TO RESEARCH QUESTIONS

The review highlights a growing interest in using CPSCs to oversee contract execution and ensure contract compliance. Not all reviewed approaches deployed smart contracts to monitor legal contracts; instead, many have utilized smart contracts to monitor and control sensor data and events and react to violations. Here are the findings of each research question and some insights about the conducted mapping review:

For **RQ1** (*What are the current platforms that support the implementation of CPSCs for event-based monitoring?*): the findings of the literature review, which are reported in section IV, demonstrate that the development, deployment, and invocation of CPSCs involve a variety of platforms, development tools, and data carriers. Table 3 presents a summary of the CPSC literature with architectural characteristics themes. The centralized, decentralized, and hybrid platforms were proposed as different CPSC approaches for compliance monitoring. Few studies proposed and explored the hybrid and centralized architecture for CPSCs, with most studies using a decentralized implementation for distributed execution of smart contracts. The decentralized approach is favored because it offers the benefits of blockchain technology, such as transparency and immutability.

Also, the results of the review indicate that smart contract execution and monitoring can occur on the blockchain, while also some parts of the smart contracts may be executed off-chain, and some monitoring procedures may remain off-chain as well. The prevalent method described in the literature for enforcing compliance is to conduct monitoring outside of smart contracts, where smart contracts can utilize feedback from external sources to respond to monitoring outcomes. In reality, monitoring using blockchain-based smart contracts can be challenging due to the high cost of execution and monitoring on the blockchain. Therefore, several approaches proposed for storing sensor data/events and conducting monitoring off-chain to reduce costs.

The literature also revealed that smart contracts are typically created manually and customized to fit the targeted programming language and platform, making their creation an exceedingly challenging task for developers in various CPS fields. Finally, the proposed layered architecture of CPSC, shown in Fig. 10, is based on a modular architecture that may consist of several distinct components, and it is the responsibility of the developers to remove, add, or modify them according to the requirements of the targeted field in CPSs.

For **RQ2** (*How do CPSCs produce and consume events from/to the outside world for monitoring?*): the results from section IV show that smart contracts cannot access and monitor sensor data and events from the outside world directly. External components and technologies are needed to verify, consume, and produce these data and events for compliance monitoring. Tables 4 and 5, together with Figs. 9 and 10, summarize the needed components, technologies, and communication patterns for consuming and producing events. An oracle is used as a third-party agent to check the veracity of sensor data and events that cannot be accessed by the smart contract or cannot be reached by the physical world. Also, additional storage could be used as a conduit for consuming and producing sensor data and events, e.g., a cloud environment could store sensor data and events collected from an oracle. Sensor data and events are either made available to anyone to ensure transparency or are only accessible to certain parties in order to preserve privacy. Three methods

TABLE 6. Mapping review limitations and related mitigation approaches.

| Limitations/Threats | Mitigation Strategies |
|--|--|
| Papers written in languages other than English, which may contain useful information on the topic, were not included. | None. |
| Having mainly one person (the first author) involved in the review, selection, and filtering of papers might bias the validity of the mapping review content. | Having multiple mapping review authors agree on the relevance of borderline papers helped reduce the risk of bias. |
| The selected databases and the query used may have left out relevant papers. | Snowballing was used as a complementary paper selection strategy. |
| Deciding the relevance of a paper based on the abstract only may result in the exclusion of relevant papers. | The irrelevant papers have been excluded after reading title, keywords, abstract, and conclusion. |
| The results of the mapping review are limited to information collected in academic publications (peer-reviewed journals and conference papers), which limits external validity and might not reflect industrial reality. | None. |

are used to store sensor data and events, including storing them directly in the blockchain, in third-party storage (off-chain), or dividing them between the blockchain and multiple third-party storage entities.

For **RQ3** (*What are current techniques for mitigating CPSC execution failures in event-based monitoring?*): the results reported in section V show that the components and technologies needed for supporting CPSCs could come with multiple types of infrastructure failures. Fig. 11 summarizes failure types and corresponding mitigation approaches. Also, the existing literature focuses more on run-time smart contract execution failures than on infrastructure failures and solutions (e.g., limitations of IoT devices). This could be a significant barrier hindering the development of smart contracts in CPSs. Therefore, infrastructure failures require further research in order to better address challenges related to integrating smart contracts with CPSs in practice.

For **RQ4** (*What are the main technical challenges faced in the development of CPSCs for event-based monitoring?*): the results presented in section VI reveal that the role of CPSCs in compliance monitoring brings multiple technical challenges. Multiple aspects need to be considered while making architectural decisions related to CPSCs. Important technical challenges relate to security, availability, robustness, privacy, and legal and regulatory aspects.

B. THREATS TO VALIDITY

As for any literature review, this mapping review is prone to several limitations and threats to validity, which are listed in Table 6 together with related mitigation strategies.

In addition, as this is a literature review, there was no experiment-based or empirical assessment of the various approaches discussed in the paper.

VIII. CONCLUSION AND FUTURE OPPORTUNITIES

This paper presented the findings of a mapping review on the use of CPSCs for compliance monitoring, with the aim to provide insights on existing architectures, patterns, infrastructure failure types, and technical challenges. To our best knowledge, this mapping review is the first literature review focused on smart contract development for event-based monitoring in cyber-physical systems.

The main conclusions of this mapping review are summarized below, together with corresponding suggestions for potential elements of solutions and future research opportunities:

- 1) Existing research on smart contract compliance monitoring in CPSs is limited to the blockchain as a dominant execution environment with back-end storage and a mediator for transferring sensor data and events because this architecture maintains secure, immutable, and redundant storage. In the future, research should be extended to cover other approaches to overcome the limitation of blockchains in terms of cost, speed, and power usage. For example, off-chain execution could be used as an alternative approach to blockchain for executing smart contracts [61], [69]. This strategy requires shifting data and/or smart contracts “off” the blockchain and to a different platform that performs better in terms of execution speed, storage, regulatory compliance, and so on. This approach can decrease blockchain costs and provide better scalability by improving transaction processing times and storage [61]. Off-chain execution, however, may jeopardize some of the benefits of security and transparency that come with using blockchain [61]. Another option to explore could involve Directed Acyclic Graph based Distributed Ledgers, which offer better scalability than blockchain at the expense of more limited security due to the absence of consensus algorithms [48].
- 2) Regarding the lack of trusted data carriers, external data and events from the physical world are necessary for smart contracts execution and monitoring. The connection between the cyber and physical worlds is made possible by reliable data carriers (e.g., oracles) to verify the data generated by physical components and deliver them to smart contracts. However, the trustworthiness and security of data carriers are seen as major impediments to the use of smart contracts within CPSs. Therefore, a robust ecosystem of reliable data carriers is necessary to improve the implementation of CPSCs. Many potential solutions could be utilized to establish a robust ecosystem to ensure the accuracy of data and events that are roaming around several physical and cyber components in various CPSC applications, including but not limited to:
 - *Data Provenance*: Tracking the source of data, including its origin, usage, and history can help minimize data tampering. Blockchain provides an immutable data record where data origins can be traceable to ensure data accuracy and consistency [51].
 - *Data Protection*: Taking necessary privacy and security measures to protect the shared ecosystem’s data and events is also something to be considered. This involves applying measures such as preventing unauthorized access or other privacy and security practices described in section VI of this paper.
 - *Data Standardization*: The use of standardized data formats would also facilitate the sharing of data and events while supporting interoperability amongst the smart contract and other systems [61].
- 3) Few studies have focused on monitoring infrastructure failures; instead, most studies have focused on failures related to the execution of smart contracts. There is a need to extend the research on infrastructure failures of CPSs due to the increased adoption of IoT, cloud, and other technologies, which is crucial for the accurate and reliable monitoring and execution of smart contracts. As mentioned previously in section V, hardware failures can occur with actuators/sensors for a variety of reasons (e.g., attacks, damages, or degradation [9]), potentially resulting in major data loss. Potential solutions aiming to reduce the impact of hardware failures could involve the use of IoT/DLT integration that supports redundancy, e.g., distributing multiple actuators/sensors across various nodes such that if one fails, another can take over the task. Such solutions can help prevent single points of failure [9].
- 4) For compliance monitoring, existing studies have often used an extra layer in the monitoring architecture between smart contracts and the external world, where usually the monitoring happens outside the smart contracts. Also, based on the monitoring decision, a suitable function within the SC will be called. There is however a need for a reference monitoring model of generated data and events to identify and deal with them efficiently.
- 5) The body of smart contract functions is often coded manually using languages such as Solidity, JavaScript, Go, and Java. There is no method for directly converting a real contract into an enforceable smart contract, which further increases the complexity of adopting smart contracts in other CPS areas. There has been some research on this matter, such as the use of domain-specific languages, template-based code generators, and verification. Template-based code generators provide templates to assist developers in creating smart contracts efficiently. These templates consist of predefined common smart contract constructs code, as proposed by Hamdaqa et al. [21]. Domain-specific languages can also represent a potential solution [20], [22], [70]. By using domain-specific languages, developers can use domain concepts, rules, and

high-level coding abstractions related to contracts, which can simplify the process of creating smart contracts, hence reducing coding errors, development time, and overall complexity. Such languages can hence help address some of the challenges discussed in our review. Verification is another potential solution to address the complexity of coding smart contracts [19]. This usually involves the use of (formal) verification methods to ensure that a smart contract complies with its intended specification (including legal obligations) and can prevent vulnerabilities. These solutions have their own challenges in terms of additional time and domain expertise needed to exploit them properly [20], [21].

- 6) Common CPSC challenges related to availability, robustness, privacy, and legal and regulatory aspects have not been investigated extensively yet, which again provides many opportunities for further research.

ACKNOWLEDGMENT

The authors thank L. Logrippo, M. Roveri, K. Nault, A. Lopes, and O. Oladosu for their feedback on earlier versions of this paper. They also acknowledge useful suggestions from the anonymous reviewers.

REFERENCES

- [1] H. Hasan, E. AlHadhrami, A. AlDhaheeri, K. Salah, and R. Jayaraman, "Smart contract-based approach for efficient shipment management," *Comput. Ind. Eng.*, vol. 136, pp. 149–159, Oct. 2019, doi: [10.1016/j.cie.2019.07.022](https://doi.org/10.1016/j.cie.2019.07.022).
- [2] A. Shukla, S. K. Mohalik, and R. Badrinath, "Smart contracts for multi-agent plan execution in untrusted cyber-physical systems," in *Proc. IEEE 25th Int. Conf. High Perform. Comput. Workshops (HiPCW)*, Dec. 2018, pp. 86–94, doi: [10.1109/HiPCW.2018.8634034](https://doi.org/10.1109/HiPCW.2018.8634034).
- [3] S. Dhairour and S. Assar, "A systematic literature review of blockchain-enabled smart contracts: Platforms, languages, consensus, applications and choice criteria," in *Research Challenges in Information Science*. Cham, Switzerland: Springer, Jun. 2020, pp. 249–266, doi: [10.1007/978-3-030-50316-1_15](https://doi.org/10.1007/978-3-030-50316-1_15).
- [4] S. Parjuangan, "Systematic literature review of blockchain based smart contracts platforms," in *Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITSI)*, Oct. 2020, pp. 381–386, doi: [10.1109/ICITSI50517.2020.9264908](https://doi.org/10.1109/ICITSI50517.2020.9264908).
- [5] E. Leka, B. Selimi, and L. Lamani, "Systematic literature review of blockchain applications: Smart contracts," in *Proc. Int. Conf. Inf. Technol. (InfoTech)*, Sep. 2019, pp. 1–3, doi: [10.1109/InfoTech.2019.8860872](https://doi.org/10.1109/InfoTech.2019.8860872).
- [6] A. Vacca, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges," *J. Syst. Softw.*, vol. 174, Apr. 2021, Art. no. 110891, doi: [10.1016/j.jss.2020.110891](https://doi.org/10.1016/j.jss.2020.110891).
- [7] S. R. Niya, S. S. Jha, T. Bocek, and B. Stiller, "Design and implementation of an automated and decentralized pollution monitoring system with blockchains, smart contracts, and LoRaWAN," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2018, pp. 1–4, doi: [10.1109/NOMS.2018.8406329](https://doi.org/10.1109/NOMS.2018.8406329).
- [8] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Proc. Design Autom. Conf.*, 2010, pp. 731–736, doi: [10.1145/1837274.1837461](https://doi.org/10.1145/1837274.1837461).
- [9] L. Hang and D.-H. Kim, "Design and implementation of an integrated IoT blockchain platform for sensing data integrity," *Sensors*, vol. 19, no. 10, p. 2228, May 2019, doi: [10.3390/s19102228](https://doi.org/10.3390/s19102228).
- [10] N. Szabo. (1997). *The Idea of Smart Contracts*. [Online]. Available: <https://bit.ly/2PXmUfz>
- [11] L. Hang and D.-H. Kim, "Reliable task management based on a smart contract for runtime verification of sensing and actuating tasks in IoT environments," *Sensors*, vol. 20, no. 4, p. 1207, Feb. 2020, doi: [10.3390/s20041207](https://doi.org/10.3390/s20041207).
- [12] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016, doi: [10.1109/ACCESS.2016.2566339](https://doi.org/10.1109/ACCESS.2016.2566339).
- [13] J. Lockl, V. Schlatt, A. Schweizer, N. Urbach, and N. Harth, "Toward trust in Internet of Things ecosystems: Design principles for blockchain-based IoT applications," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1256–1270, Nov. 2020, doi: [10.1109/TEM.2020.2978014](https://doi.org/10.1109/TEM.2020.2978014).
- [14] T. R. P. M. Rúbio, Z. Kokkinogenis, H. L. Cardoso, R. J. F. Rossetti, and E. Oliveira, "Regulating blockchain smart contracts with agent-based markets," in *Proc. EPIA Conf. Artif. Intell.* Cham, Switzerland: Springer, 2019, pp. 399–411, doi: [10.1007/978-3-030-30241-2_34](https://doi.org/10.1007/978-3-030-30241-2_34).
- [15] E. Solaiman, T. Wike, and I. Sfyarakis, "Implementation and evaluation of smart contracts using a hybrid on- and off-blockchain architecture," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 1, pp. 1–12, Jan. 2021, doi: [10.1002/cpe.5811](https://doi.org/10.1002/cpe.5811).
- [16] V. Buterin. (2013). *Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform*. [Online]. Available: <https://github.com/ethereum/wiki/wiki/White-Paper>
- [17] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: Platforms, applications, and design patterns," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Mar. 2017, pp. 1–4, doi: [10.1007/978-3-319-70278-0_31](https://doi.org/10.1007/978-3-319-70278-0_31).
- [18] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, and D. Enyeart, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–13, doi: [10.1145/3190508.3190538](https://doi.org/10.1145/3190508.3190538).
- [19] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019, doi: [10.1109/TSMC.2019.2895123](https://doi.org/10.1109/TSMC.2019.2895123).
- [20] M. Wöhrer and U. Zdun, "Domain specific language for smart contract development," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2020, pp. 1–9, doi: [10.1109/ICBC48266.2020.9169399](https://doi.org/10.1109/ICBC48266.2020.9169399).
- [21] M. Hamdaqa, L. A. P. Met, and I. Qasse, "iContractML 2.0: A domain-specific language for modeling and deploying smart contracts onto multiple blockchain platforms," *Inf. Softw. Technol.*, vol. 144, Apr. 2022, Art. no. 106762, doi: [10.1016/j.infsof.2021.106762](https://doi.org/10.1016/j.infsof.2021.106762).
- [22] S. Sharifi, A. Parvizmosaed, D. Amyot, L. Logrippo, and J. Mylopoulos, "Symbiole: Towards a specification language for legal contracts," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 364–369, doi: [10.1109/RE48521.2020.00049](https://doi.org/10.1109/RE48521.2020.00049).
- [23] C. Okoli, "A guide to conducting a standalone systematic literature review," *Commun. Assoc. Inf. Syst.*, vol. 37, pp. 1–13, Jan. 2015, doi: [10.17705/1CAIS.03743](https://doi.org/10.17705/1CAIS.03743).
- [24] N. R. Haddaway, M. J. Page, C. C. Pritchard, and L. A. McGuinness, "PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis," *Campbell Systematic Rev.*, vol. 18, no. 2, e1230, 2022, doi: [10.1002/cj.1230](https://doi.org/10.1002/cj.1230).
- [25] D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, "Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement," *BMJ*, vol. 339, no. 1, p. b2535, Jul. 2009, doi: [10.1136/bmj.b2535](https://doi.org/10.1136/bmj.b2535).
- [26] E. Mourão, J. F. Pimentel, L. Murta, M. Kalinowski, E. Mendes, and C. Wohlin, "On the performance of hybrid search strategies for systematic literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 123, Jul. 2020, Art. no. 106294, doi: [10.1016/j.infsof.2020.106294](https://doi.org/10.1016/j.infsof.2020.106294).
- [27] Covidence. (2021). *Covidence. Better Systematic Review Management*. [Online]. Available: <https://www.covidence.org/>
- [28] K. Guler, D. Beyer, and C. Santos, "Computer-implemented method for automatic contract monitoring," U.S. Patent 10/443 930, Jan. 20, 2005. [Online]. Available: <https://patents.google.com/patent/US20050015319A1/en>
- [29] M. Dawson, "Smart contract compliance monitoring and enforcement," U.S. Patent 16/176 843, May 9, 2020. [Online]. Available: <https://patents.google.com/patent/US20200134711A1/en>
- [30] D. Rice, "Method and system for monitoring a smart contract on a distributed ledger," U.S. Patent 16/019 203, Dec. 26, 2019. [Online]. Available: <https://patents.google.com/patent/US20190392178A1/en>
- [31] A. Bagozi, D. Bianchini, V. De Antonellis, M. Garda, and M. Melchiori, "Exploiting blockchain and smart contracts for data exploration as a service," in *Proc. 21st Int. Conf. Inf. Integr. Web Appl. Services*, Dec. 2019, p. 393, doi: [10.1145/3366030.3366075](https://doi.org/10.1145/3366030.3366075).

- [32] O. López-Pintado, M. Dumas, L. García-Bañuelos, and I. Weber, "Interpreted execution of business process models on blockchain," in *Proc. IEEE 23rd Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, Oct. 2019, pp. 206–215, doi: [10.1109/EDOC.2019.00033](https://doi.org/10.1109/EDOC.2019.00033).
- [33] S. Tahmasebi, J. Habibi, and A. Shamsaie, "A scalable architecture for monitoring IoT devices using ethereum and fog computing," in *Proc. 4th Int. Conf. Smart City, Internet Things Appl. (SCIOT)*, Sep. 2020, pp. 66–76, doi: [10.1109/SCIOT50840.2020.9250193](https://doi.org/10.1109/SCIOT50840.2020.9250193).
- [34] P. Kumar, R. Kumar, G. P. Gupta, R. Tripathi, A. Jolfaei, and A. K. M. N. Islam, "A blockchain-orchestrated deep learning approach for secure data transmission in IoT-enabled healthcare system," *J. Parallel Distrib. Comput.*, vol. 172, pp. 69–83, Feb. 2023, doi: [10.1016/j.jpdc.2022.10.002](https://doi.org/10.1016/j.jpdc.2022.10.002).
- [35] A. Alzubaidi, K. Mitra, and E. Solaiman, "Smart contract design considerations for SLA compliance assessment in the context of IoT," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2021, pp. 74–81, doi: [10.1109/SmartIoT52359.2021.00021](https://doi.org/10.1109/SmartIoT52359.2021.00021).
- [36] F. Jamil, N. Iqbal, S. Ahmad, and D. Kim, "Peer-to-peer energy trading mechanism based on blockchain and machine learning for sustainable electrical power supply in smart grid," *IEEE Access*, vol. 9, pp. 39193–39217, 2021, doi: [10.1109/ACCESS.2021.3060457](https://doi.org/10.1109/ACCESS.2021.3060457).
- [37] L. Hang, I. Ullah, and D.-H. Kim, "A secure fish farm platform based on blockchain for agriculture data integrity," *Comput. Electron. Agricult.*, vol. 170, Mar. 2020, Art. no. 105251, doi: [10.1016/j.compag.2020.105251](https://doi.org/10.1016/j.compag.2020.105251).
- [38] K. Wright, M. Martinez, U. Chadha, and B. Krishnamachari, "SmartEdge: A smart contract for edge computing," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber. Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1685–1690, doi: [10.1109/Cybermatics_2018.2018.00281](https://doi.org/10.1109/Cybermatics_2018.2018.00281).
- [39] N. Neidhardt, C. Köhler, and M. Nüttgens, "Cloud service billing and service level agreement monitoring based on blockchain," *EMISA Forum*, vol. 18, no. 1, pp. 65–69, 2018. [Online]. Available: <http://ceur-ws.org/Vol-2097/paper11.pdf>
- [40] H. Zhou, C. de Laat, and Z. Zhao, "Trustworthy cloud service level agreement enforcement with blockchain based smart contract," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2018, pp. 255–260, doi: [10.1109/CloudCom2018.2018.00057](https://doi.org/10.1109/CloudCom2018.2018.00057).
- [41] A. Smirnov and N. Teslya, "Robot interaction through smart contract for blockchain-based coalition formation," in *Product Lifecycle Management to Support Industry 4.0*. Cham, Switzerland: Springer, 2018, pp. 611–620, doi: [10.1007/978-3-030-01614-2_56](https://doi.org/10.1007/978-3-030-01614-2_56).
- [42] X. Liu, K. Muhammad, J. Lloret, Y.-W. Chen, and S.-M. Yuan, "Elastic and cost-effective data carrier architecture for smart contract in blockchain," *Future Gener. Comput. Syst.*, vol. 100, pp. 590–599, Nov. 2019, doi: [10.1016/j.future.2019.05.042](https://doi.org/10.1016/j.future.2019.05.042).
- [43] R. B. Uriarte, H. Zhou, K. Kritikos, Z. Shi, Z. Zhao, and R. De Nicola, "Distributed service-level agreement management with smart contracts and blockchain," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 14, pp. 1–17, Jul. 2021, doi: [10.1002/cpe.5800](https://doi.org/10.1002/cpe.5800).
- [44] H. A. Breiki, L. A. Qassem, K. Salah, M. H. U. Rehman, and D. Sevtnovic, "Decentralized access control for IoT data using blockchain and trusted oracles," in *Proc. IEEE Int. Conf. Ind. Internet (ICII)*, Nov. 2019, pp. 248–257, doi: [10.1109/ICII.2019.00051](https://doi.org/10.1109/ICII.2019.00051).
- [45] J. Ladleif, I. Weber, and M. Weske, "External data monitoring using oracles in blockchain-based process execution," in *Business Process Management: Blockchain and Robotic Process Automation Forum*. Cham, Switzerland: Springer, 2020, pp. 67–81, doi: [10.1007/978-3-030-58779-6_5](https://doi.org/10.1007/978-3-030-58779-6_5).
- [46] C. Dienbauer, B. Pittl, W. Mach, and E. Schikuta, "A penalty-aware cloud monitoring system based on blockchains," in *Proc. 22nd Int. Conf. Inf. Integr. Web Appl. Services*, Nov. 2020, pp. 1–19, doi: [10.1145/3428757.3429130](https://doi.org/10.1145/3428757.3429130).
- [47] M. Taghavi, J. Bentahar, H. Otrok, and K. Bakhtiyari, "A blockchain-based model for cloud service quality monitoring," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 276–288, Mar. 2020, doi: [10.1109/TSC.2019.2948010](https://doi.org/10.1109/TSC.2019.2948010).
- [48] C. Lehnert, G. Engel, and T. Greiner, "Distributed ledger and smart contract based approach for IoT sensor applications," in *Proc. Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2020, pp. 1–6, doi: [10.1109/COINS49042.2020.9191409](https://doi.org/10.1109/COINS49042.2020.9191409).
- [49] P. Kochovski, V. Stankovski, S. Gec, F. Faticanti, M. Savi, D. Siracusa, and S. Kum, "Smart contracts for service-level agreements in edge-to-cloud computing," *J. Grid Comput.*, vol. 18, no. 4, pp. 673–690, Dec. 2020, doi: [10.1007/s10723-020-09534-y](https://doi.org/10.1007/s10723-020-09534-y).
- [50] F. Jamil, S. Ahmad, N. Iqbal, and D.-H. Kim, "Towards a remote monitoring of patient vital signs based on IoT-based blockchain integrity management platforms in smart hospitals," *Sensors*, vol. 20, no. 8, p. 2195, Apr. 2020, doi: [10.3390/s20082195](https://doi.org/10.3390/s20082195).
- [51] G. Baralla, A. Pinna, R. Tonelli, M. Marchesi, and S. Ibba, "Ensuring transparency and traceability of food local products: A blockchain application to a smart tourism region," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 1, p. e5857, Jan. 2021, doi: [10.1002/cpe.5857](https://doi.org/10.1002/cpe.5857).
- [52] D. Cacciagrano, F. Corradini, G. Mazzante, L. Mostarda, and D. Sestili, "Off-chain execution of IoT smart contracts," in *Advanced Information Networking and Applications*. Cham, Switzerland: Springer, 2021, pp. 608–619, doi: [10.1007/978-3-030-75075-6_50](https://doi.org/10.1007/978-3-030-75075-6_50).
- [53] M. Shukla, J. Lin, and O. Seneviratne, "BlockIoT-RETEL: Blockchain and IoT based read-execute-transact-erase-loop environment for integrating personal health data," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Dec. 2021, pp. 237–243, doi: [10.1109/Blockchain53845.2021.00039](https://doi.org/10.1109/Blockchain53845.2021.00039).
- [54] F. Jamil, M. Ibrahim, I. Ullah, S. Kim, H. K. Kahng, and D.-H. Kim, "Optimal smart contract for autonomous greenhouse environment based on IoT blockchain network in agriculture," *Comput. Electron. Agricult.*, vol. 192, Jan. 2022, Art. no. 106573, doi: [10.1016/j.compag.2021.106573](https://doi.org/10.1016/j.compag.2021.106573).
- [55] M. Pustišek, M. Chen, A. Kos, and A. Kos, "Decentralized machine autonomy for manufacturing servitization," *Sensors*, vol. 22, no. 1, p. 338, Jan. 2022, doi: [10.3390/s22010338](https://doi.org/10.3390/s22010338).
- [56] T. Hewa, A. Braeken, M. Liyanage, and M. Ylianttila, "Fog computing and blockchain-based security service architecture for 5G industrial IoT-enabled cloud manufacturing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7174–7185, Oct. 2022, doi: [10.1109/TII.2022.3140792](https://doi.org/10.1109/TII.2022.3140792).
- [57] C. Zhang, L. Zhu, and C. Xu, "BSDP: Blockchain-based smart parking for digital-twin empowered vehicular sensing networks with privacy protection," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 7237–7246, May 2023, doi: [10.1109/TII.2022.3223211](https://doi.org/10.1109/TII.2022.3223211).
- [58] G. Falazi, A. Lamparelli, U. Breitenbuecher, F. Daniel, and F. Leymann, "Unified integration of smart contracts through service orientation," *IEEE Softw.*, vol. 37, no. 5, pp. 60–66, Sep. 2020, doi: [10.1109/MS.2020.2994040](https://doi.org/10.1109/MS.2020.2994040).
- [59] O. Hornýák and G. F. Alkhoury, "Smart contracts in the automotive industry," in *Vehicle and Automotive Engineering*. Cham, Switzerland: Springer, 2020, pp. 148–157, doi: [10.1007/978-981-15-9529-5](https://doi.org/10.1007/978-981-15-9529-5).
- [60] S. S. Arumugam, V. Umashankar, N. C. Narendra, R. Badrinath, A. P. Mujumdar, J. Holler, and A. Hernandez, "IoT enabled smart logistics using smart contracts," in *Proc. 8th Int. Conf. Logistics, Informat. Service Sci. (LISS)*, Aug. 2018, pp. 1–6, doi: [10.1109/LISS.2018.8593220](https://doi.org/10.1109/LISS.2018.8593220).
- [61] 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, doi: [10.1007/s10506-018-9223-3](https://doi.org/10.1007/s10506-018-9223-3).
- [62] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, N. Kumar, and M. M. Hassan, "A privacy-preserving-based secure framework using blockchain-enabled deep-learning in cooperative intelligent transport system," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16492–16503, Sep. 2022, doi: [10.1109/TITS.2021.3098636](https://doi.org/10.1109/TITS.2021.3098636).
- [63] R. Kumar, P. Kumar, M. Aloqaily, and A. Aljuhani, "Deep-learning-based secure zero touch networks," *IEEE Commun. Mag.*, vol. 61, no. 2, pp. 96–102, Feb. 2023, doi: [10.1109/MCOM.001.2200294](https://doi.org/10.1109/MCOM.001.2200294).
- [64] D. Das, S. Banerjee, P. Chatterjee, U. Ghosh, U. Biswas, and W. Mansoor, "Security, trust, and privacy management framework in cyber-physical systems using blockchain," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2023, pp. 1–6, doi: [10.1109/CCNC51644.2023.10060483](https://doi.org/10.1109/CCNC51644.2023.10060483).
- [65] R. Kumar, P. Kumar, A. Aljuhani, A. K. M. N. Islam, A. Jolfaei, and S. Garg, "Deep learning and smart contract-assisted secure data sharing for IoT-based intelligent agriculture," *IEEE Intell. Syst.*, early access, Aug. 31, 2022, doi: [10.1109/MIS.2022.3201553](https://doi.org/10.1109/MIS.2022.3201553).
- [66] B. D. Deebak and F. Al-Turjman, "A robust and distributed architecture for 5G-enabled networks in the smart blockchain era," *Comput. Commun.*, vol. 181, pp. 293–308, Jan. 2022, doi: [10.1016/j.comcom.2021.10.015](https://doi.org/10.1016/j.comcom.2021.10.015).
- [67] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "HAWK: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 839–858, doi: [10.1109/SP.2016.55](https://doi.org/10.1109/SP.2016.55).

- [68] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 270–282, doi: [10.1145/2976749.2978326](https://doi.org/10.1145/2976749.2978326).
- [69] C. Molina-Jimbenez, E. Solaiman, I. Sfyarakis, I. Ng, and J. Crowcroft, "On and off-blockchain enforcement of smart contracts," in *Proc. Euro-Par Int. Workshops*, Jan. 2019, pp. 342–354, doi: [10.1007/978-3-030-10549-5_27](https://doi.org/10.1007/978-3-030-10549-5_27).
- [70] A. Rasti, D. Amyot, A. Parvizimosaed, M. Roveri, L. Logrippo, A. A. Anda, and J. Mylopoulos, "Symboleo2SC: From legal contract specifications to smart contracts," in *Proc. 25th Int. Conf. Model Driven Eng. Lang. Syst.*, New York, NY, USA, Oct. 2022, pp. 300–310, doi: [10.1145/3550355.3552407](https://doi.org/10.1145/3550355.3552407).



SOFANA ALFUHAID received the M.Sc. degree in digital transformation and innovation (DTI) from the University of Ottawa, Canada, in 2020. She is currently pursuing the Ph.D. degree in DTI, working on the generation of smart contracts from Symboleo specifications of legal contracts. Her thesis focused on blockchain-based traceability. She was a Teaching Assistant with King Abdulaziz University, Saudi Arabia, from 2012 to 2017. Her research interests include blockchains, smart

contracts, cyber-physical systems, non-functional requirements, and domain-specific languages.



DANIEL AMYOT (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Ottawa, Canada, in 2002. He led the standardization of the user requirements notation with the International Telecommunication Union, from 2002 to 2013. He is currently a Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. He also co-leads a project on the Symboleo specification language for legal contracts. His research

interests include requirements engineering, software engineering, goal and process modeling, regulatory compliance, cyber-physical systems, smart contracts, and healthcare informatics. He was the General Chair of the IEEE International Requirements Engineering Conference, in 2015, and the Program Co-Chair, in 2018. He is also on the editorial boards of *SoSyM* and the *Requirements Engineering*.



AMAL AHMED ANDA (Member, IEEE) received the Ph.D. degree in computer science from the University of Ottawa, in 2020, on a scholarship from the Libyan Ministry of Education. She was an Assistant Professor with Tripoli University. In Libya, she contributed to complex information systems, including large databases (students, employees, teachers, and retirees at the national level) and financial systems. She is currently a Postdoctoral Fellow with the University of Ottawa

and a part-time Professor with the Algonquin College and the St. Lawrence College, Canada. Her research interests include model-driven software engineering, requirements engineering, adaptive cyber-physical systems, and smart contracts.



JOHN MYLOPOULOS received the Ph.D. degree from Princeton University, in 1970. He joined the Faculty of the Department of Computer Science, University of Toronto, Canada, in 1970. He was the Project Leader for a project titled Lucretius: Foundations for Software Evolution, funded by an Advanced Grant from the European Research Council (2011–2016). He holds a Professor Emeritus position at the Universities of Toronto (Canada) and Trento (Italy), and he is working at the University of Ottawa on a project titled Engineering Smart Contracts as a Visiting

Researcher. His research interests include conceptual modeling, requirements engineering, data semantics, and knowledge management. He is a fellow of the Association for the Advancement of Artificial Intelligence (AAAI) and the Royal Society of Canada (Academy of Applied Sciences). He has served as the Program Chair/General Chair for international conferences in artificial intelligence, databases, and software engineering, including IJCAI (1991), Requirements Engineering (1997, 2011), and VLDB (2004).

• • •