(2025) 35(2) JUUM 76 – 95
https://doi.org/10.17576/juum-2025-3502-05

SMART CONTRACTS FROM CODING TO EXECUTION

[1]Ra'ed Fawzi Aburoub*
[2]Nabeel Mahdi Althabhawi
[3]Mohamad Rizal Abd Rahman
[4]Ammar Abbas Kadhim

[1]Postgraduate Student (PhD), Faculty of Law, Universiti Kebangsaan Malaysia
[2]Senior Lecturer, Faculty of Law, Universiti Kebangsaan Malaysia
[3]Associate Professor, Faculty of Law, Universiti Kebangsaan Malaysia
[4]Professor, Department of Law, Al-Mustaqbal University

Corresponding author: p126484@siswa.ukm.edu.my*

ABSTRACT

This paper explores the lifecycle of a smart contract, from the stages of coding and deployment to execution and verification, in order to show that a smart contract can indeed be self-executing, transparent, and immutable. While such functionalities introduce efficiency, trust, and reliability within industries such as financial, supply chain management, and health sectors, smart contracts at the same time have a host of technical and legal challenges arising. This paper identifies key issues: critical vulnerabilities in coding, deployment on immutable blockchains, address assignment complexities, triggering mechanisms, and aspects of privacy. This study has adopted a critical analytical approach to evaluate the technical and legal aspects of smart contract formation, complemented by inductive reasoning to derive general insights and recommendations from specific cases and patterns. The study states that the apt legal framework must be provided for liability, regulatory compliance, and solutions that would be unlooked-for. It further supports hybrid models that blend automation with human oversight, superior communication protocols regarding updating an address, and the use of technologies that allow transparency with the preservation of confidentiality in a balance. The concrete ideas it offers are attempts at technology design aligned with legal frameworks by bringing developers, regulators, and stakeholders together in implementing certain solutions. It emphasizes that continuous research will hence be important to assure reliability, security, and equitability in the adoption of smart contracts, expanding possibilities for their application in an increasingly changing digital environment.

Keywords: Smart contracts; blockchain; coding; deployment; execution.

INTRODUCTION

Smart contracts are an agreement that executes autonomously based on its terms when all conditions are met. Conceived in the 1990s by Nick Szabo, a combination of contract law with computer programming, they finally found an application that facilitates trustless transactions on blockchain networks (Szabo, 1996). The great popularity of these contracts, especially in their perceived capability to cut off the intermediaries, has made them gain widespread acceptance due to their low costs and enhanced efficiency; hence, this represents one of the cornerstones of digital ecosystems and decentralized applications (Christidis & Devetsikiotis, 2016). Their applications extend from finance to supply chain management to healthcare, among others, thus being versatile and having a potentially transformative impact (Pereira et al., 2020).

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

This study focuses on addressing the technical and legal aspects of the lifecycle of smart contracts despite their potential and advantages over traditional contracts. While smart contracts promise self-execution, transparency and immutability but face issues like coding vulnerabilities, complexity in deployment, address assignment, triggering mechanisms and privacy concerns. And they are rigid and not adaptable in unexpected scenarios hence we need a hybrid approach of automation and human oversight. The study identifies a gap in aligning technology with the legal framework to ensure liability, regulatory compliance and balanced solutions hence we need developers, regulators and stakeholders to work together to address these challenges.

This paper investigates the lifecycle of smart contracts, from coding and deployment to execution and verification. The article explains each step in detail and de-mystifies how these contracts work, showing their transformative potential in a variety of industries. The journey will be important in effectively leveraging smart contracts in current and emerging use cases. This paper has analyzed, in view of operational challenges, the legal effect which the formation process of smart contracts could have and provides an understanding of limitations further in ways of development.

The blockchain technology is the backbone of smart contracts, thus enabling them to work in an automated and see-through manner. The technology runs on distributed ledgers that ensure tamper-resistant execution and immutable records as part of the solution to the question of trust in traditional ways of enforcing contracts. Transparency and verifiability via a blockchain network also give confidence to users and other stakeholders in the smart contracts (Buterin, 2013). This is where the consensus mechanism of this technology guarantees that participants in a network agree to something, further setting the integrity of smart contracts in stone (Imteaj, et al., 2021). Innovations like sharding and layer-two solutions in evolving blockchain platforms promise scalability and efficiency for the broadening of smart contract adoption (Timuçin & Biroğul, 2023).

Three main features constitute the basis on which smart contracts stand out: self-execution, transparency, and being tamper-proof. Due to self-execution, upon the satisfaction of the defined conditions, the contract comes into effect all by its own means without the influence or interference of any other intervening agent. This greatly reduces delays and the ability of humans to commit an error (Worley & Skjellum, 2019). In that blockchain serves on an open ledger, this allows transparency in the executed visibility of the contract, instilling a sense of trust and responsibility among both parties (Christidis & Devetsikiotis, 2016). Immutability ensures, through the deployment of a smart contract, that neither the code nor the database of that contract can be tampered with and hence the integrity of the agreement is retained (Fröwis & Böhme, 2017). The features put together form the backbone to ensure reliability and efficiency within smart contracts, thus making it a disruptive innovation in the contractual framework (Vasiu & Vasiu, 2023).

Smart Contracts, characterized by their self-executing code and reliance on decentralized blockchain technology, stand in stark contrast to traditional contracts, which traditionally rely on human interpretation and often involve intermediaries for enforcement (Savelyev, 2017). Meanwhile, electronic contracts find themselves in the middle ground, leveraging digital formats without the inherent programmability of smart contracts (Loddo et al, 2022). Additionally, a fundamental distinction arises in the realm of trust and intermediaries, as smart contracts tailored for a trustless environment obviate the need for intermediaries and depend on decentralized blockchain technology for

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

transparency and immutability, while traditional contracts frequently employ intermediaries such as banks or legal entities to establish trust and enforce agreements (Giancaspro, 2017). In the meantime, Conventional contracts have humans—for example, lawyers and notaries—for writing, enforcement, and settlement, which is usually time-wasting and expensive. In smart contracts, execution is self-running, without intermediators; as a result, transaction costs come down drastically (Savelyev, 2017). However, despite that electronic contracts can enhance communication but might still necessitate intermediaries also for enforcement (Milosevic et al, 2002). The differentiation extends to the coding and programmability inherent in these contracts. Smart contracts are commonly scripted in programming languages expressly crafted for their creation, with Solidity being one of the prevalent languages for coding smart contracts on the Ethereum blockchain. The coding process entails specifying the terms, conditions, and logic governing the contract's execution. In contrast, traditional contracts, typically articulated in natural language, (like Arabic or English language) lack this programmable attribute (Hwang. et al, 2022). As for electronic contracts, they are executed similarly to traditional contracts but within an electronic environment (computers, e-mails, e-messages). However, electronic contracts, despite their digital nature, may not exhibit the same degree of programmability as smart contracts (Tok & Tunca, 2023). In the meanwhile, smart contracts are written under strict programming codes and unalterable transaction records. Once activated, a smart contract's execution cannot be halted (Nugraheni et al, 2022). In contrast, traditional contracts lack this inherent transparency and immutability, relying on amendments that may not be immediately apparent, so that traditional contracts can be amended through mutual consent, judicial decisions, or legal stipulations (Qutieshat et al, 2022). As for electronic contracts, while they offer transparency, they might not possess the immutability inherent in smart contracts. This is because electronic contracts are typically stored on central servers to authenticate electronic written documents (Hasan, 2007). On the other hand, smart contracts are inherently entwined with blockchain platforms, leveraging decentralized and distributed ledgers for execution. This integration ensures a transparent and tamper-resistant record of transactions (Stampernas, 2018). In contrast, traditional contracts typically depend on centralized databases and authorities for documentation and enforcement . This process relies on duplicates of paper records stored within these central entities, and all parties involved in the transaction are required to maintain physical copies of the contracts. Moreover, this traditional documentation method is susceptible to various risks such as damage, loss, forgery, and tampering (El Sayed El Gendy, 2019). As for electronic contracts, while digital, are versatile in their platform existence, with the capacity to operate on various platforms, including centralized databases, without mandatorily relying on blockchain infrastructure (Hasan, 2007).

METHODOLOGY

This study used a non-doctrinal (empirical) approach, looking at practical challenges, specific cases and patterns around the lifecycle of smart contracts. It used critical and inductive reasoning to evaluate technical vulnerabilities, legal challenges and hybrid solutions, real world applications and interdisciplinary collaboration. These are all hallmarks of a non-doctrinal approach which goes beyond theoretical or legal analysis and incorporates practical, technical and empirical insights into the research.

Given the pace of change in blockchain technology and its impact across industries like finance, supply chain and healthcare this approach is particularly

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

relevant. The critical and inductive analysis not only reveals the existing challenges but also enables the development of new strategies that combine the strengths of smart contracts with flexible legal frameworks to make them work in complex real world scenarios.

FORMATION PROCESS

The term "formation" in the context of smart contracts refers to the process by which these digital agreements are created, deployed, and become active on a blockchain (Durovic & Janssen, 2019). The formation of smart contracts therefore involves several key steps as follows:

CODING AND PROGRAMMING

Any software, including a smart contract, starts with choosing an appropriate programming language. Since these contracts run on various platforms, several languages prevail to date. Among these widely used is Solidity-a language specifically designed to develop smart contracts running on the Ethereum blockchain (Wood, 2014). Most likely, Solidity syntax is quite alike compared to JavaScript; that's why it is easy for web developers who are into their familiar programming to get work done. Besides Solidity, which serves the highest number of Ethereum-based development, other languages also exist-let's mention Vyper, designed to be as simple and safe, while Rust is used with its target for Solana-based smart contracts, acting among many different preferences on so many different blockchains (Ethereum Foundation, n.d).

Writing a smart contract essentially means describing the terms, conditions, and execution logic of the contract in code (Hwang. et al, 2022). These contractual obligations are translated by the developers and transformed into algorithms, defining under what and when certain actions are to be triggered (Bassan & Rabitti, 2024). A basic smart contract for crowdfunding, for example, would have conditions on fund collection, deadlines, and refund mechanisms in case the funding goal is not achieved (Sindhavad et al., 2023). This requires great attention to detail so that the code actually reflects what is intended within the agreement (Shein. Esther, 2021).

However, coding in smart contracts is not without great challenges. Bugs in the code might lead to vulnerabilities that hackers might take advantage of, resulting often in severe financial and reputational consequences. A highly infamous example that comes into the picture has to do with the hacking of DAO in 2016 due to a flaw within the code; the hackers were consequently able to drain about 60 million dollars' value in Ether. This dampened the trust placed in the workability of the Ethereum ecosystem, eventually resulting in its hard fork along the blockchain to ensure the funds were recovered (Atzei et al., 2017). Another remarkable incident can be considered to be the bug in the Parity Wallet in 2017, where a bug in this multi-signature wallet froze and made inaccessible more than $150 million in Ether. That was because of the incorrect initialization of the wallet contract, where again the accurate coding and testing practices became important (Grishchenko et al., 2018).

To mitigate such risks, developers must rigorously test and audit their smart contracts using advanced tools like MythX, which provides security analysis for Ethereum smart contracts, and OpenZeppelin's security libraries, which offer pre-audited and secure implementations of common smart contract standards (Restack, 2023). Because it comes with a set of reusable tools, such tools spot vulnerabilities like reentrancy attacks, integer overflows, and access control defects before deployment. Formal verification was also another important method developed for this purpose, using standard industrial

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

practices for ensuring strong smart contracts (Zhou et al., 2022).

The immutable nature of the blockchain amplifies these problems because, after a smart contract has been deployed, the only way to correct any kind of mistake are to deploy another contract or use some complex governance mechanism (Singh et al., 2023). Such immutability underlines that thorough pre-deployment testing and auditing are paramount (Andesta et al., 2019). In addition to integrate features such as upgradable contracts and kill switches—when legally and technically viable—they would be extra security against unexpected vulnerabilities (Seneviratne. Oshani, 2024). As smart contracts increase in terms of complexity, their keys to success will come through balancing functionality, security, and efficiency (Zhou et al., 2022). These measures, as taken together, ensure increasing the reliability of smart contracts and pave the way to broader adoption in a secure and trustworthy manner.

DEPLOYMENT ON BLOCKCHAIN

Deploying a smart contract onto the blockchain is a highly critical process that ensures it will be functional and prepared for execution on the network. First, the source code of the contract written in a high-level programming language, usually like Solidity, is compiled into bytecode by tools such as the Solidity Compiler (SOLC). This is necessary bytecode because it changes the human-target code into an understandable and executable format for EVM or any other blockchain virtual machine (Ethereum Foundation, n.d). This step ensures that the smart contract is compatible with the underlying blockchain architecture and ready to interact with other components of the system.

Once the bytecode has been generated, the next step involves deploying the contract on the blockchain. A transaction has to be sent to the blockchain network with the compiled bytecode and the initialization function if needed. Optimizing the bytecode of a contract by incorporating gas-efficient programming practices is rather important since it reduces the costs of deploying it (Chen et al., 2018). The transaction should be signed with the private key of the deploying party for authenticity and ownership purposes (Dannen, 2017). Besides that, developers should take into consideration, before deployment, the gas fees paid in cryptocurrency to compensate miners or validators for the computational resources used to store and execute the contract (Signer, 2018). The gas estimation tools, such as Ethers.js and Web3.js, would help in predicting the cost of such operations and ensuring that the deployment of a contract does not exceed the budget (Semnani & Yang, 2024).

Once compiled, the bytecode is then embedded in a blockchain transaction created by the creator of the contract (Brent et al., 2018). The transaction takes a number of required parameters, including gas fees paid in the blockchain's native cryptocurrency, the creator's address, and any initialization data that the contract requires (Zarir, et al., 2021). These parameters make sure the blockchain can process the transaction and dedicate resources to execute it (Pacheco et al., 2022). It is broadcast on the blockchain network, from where miners or validators confirm the transaction and then put it in a block (Angraal et al., 2017). Upon the confirmation from the network that it is successful, the smart contract is deployed and assigned an address (Kuppuswamy & Kodavali, 2024). The address is meaningful because this would be what is to be used by the user, applications, or other smart contracts in communicating with the deployed contract (Oluwatosin. Serah, 2023).

Deployment is an area that comes with its own set of challenges, requiring great attention to detail in both technical and

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

strategic respects. First, there is the very important issue of the accuracy and robustness of the code in a smart contract. Since any blockchain transaction is immutable, errors in the deployed code are permanent and cannot easily be fixed. Such defects may lead to unintended consequences, ranging from huge monetary losses to operational disruption or even the exploitation of the system by malicious parties. Examples of this are the DAO hack in 2016 due to a reentrancy bug in the smart contract code, which led to several million dollars in losses, consequently showing how terrible bugs could be if left unsorted (Atzei et al., 2017). With the help of Remix, a kind of online IDE for smart contracts writing and debugging, developers deal with this issue, and also Truffle provides a fully-featured suite to test and deploy contracts in controlled environments (Sharma, 2020).

Another significant challenge at deployment is the management of gas fees, which is a mechanism paying miners or validators for the computational resource usages. Gas fluctuates depending on network congestion and the complexity of a smart contract's byte code. In that respect, excessive gas could be a huge bottleneck to smaller developers or resource-poor projects, eventually hindering the effective exploitation of blockchain technology. (Oliva & Hassan, 2021). In mitigation, the developers use tools like Ethers.js and Web3.js to estimate the gas costs accurately, besides optimizing the contract code for reduced computational complexity. The strategies that could help mitigate this are deploying contracts at times of low congestion or on blockchains that have less congestion (Semnani & Yang, 2024).

Besides these technical issues, deployment challenges also involve ensuring compatibility with existing blockchain ecosystems and regulatory frameworks (Wood, 2016). For example, a developer must check whether the smart contract meets all the standards of a given blockchain that it is about to be deployed on; otherwise, in the case of deploying an Ethereum-based smart contract, it would not be ERC-20 or ERC-721 compliant (Nelaturu et al., 2023).

The verification and validation of the deployed contract constitute another vital phase. Most developers do the source code verification of their smart contracts on platforms such as Etherscan, ensuring that it is visible and can be audited by users themselves (Godoy et al., 2022). The functionality of a smart contract is usually implemented and declared in its source code, thus available for review (Ahrendt et al., 2018). Testing should, when possible, first be performed in a controlled environment such as testnets—like Rinkeby or Goerli for Ethereum—which simulate blockchain conditions with limited financial risk, to detect and fix problems before real funds are put into a mainnet (Dwyer, 2023).

Furthermore, post-deployment monitoring is also very critical. These tools, Tenderly and Hardhat, can easily track the performance and usage of the contract against what it was designed for (Khan et al., 2018). Deployment is not only a technical issue but also a strategic process since one has to ensure that compatibility with the existing infrastructure, scalability, and legal and regulatory implications are duly taken into consideration while planning and executing such deployment (Fischer et al., 2017).

Finally, smart contract to be deployed should be capable of integrating well with other blockchain components. This will include compatibility checks against DApps and ensuring that the contract can handle all transactions without performance degradation (Rashid & Siddique, 2019). Such comprehensive addressing of these various challenges will surely make smart contracts more reliable and available, thus paving the way for the

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

wider diffusion of blockchain technology in different applications.

ADDRESS ASSIGNMENT AND INTERACTION

Smart contracts are assigned their unique addresses during deployment. The blockchain network assigns this address in the transaction that confirms the creation of the smart contract (Knecht & Stiller, 2022). This address serves as the identity of the smart contract on the distributed ledger of the blockchain, hence allowing users and other contracts to locate and interact with it (Hu et al., 2018). This is an automated assignment, through the cryptographic hashing of the deployment transaction, which ensures that each address is unique and secure (Agrawal et al., 2016).

Basically, these addresses mean a great deal for entrance to interaction. The address of a smart contract lets every user send various transactions to that address to invoke internal functions or fetch data hosted on that contract (Albert et al., 2020). A typical use is in a lending protocol: For example, to use the DeFi application, one needs to interact with a smart contract for deposits and loans in consideration of its address referencing. These addresses play a big role in ensuring that all interactions are channeled to the right contract for efficient processing (Seitenov & Smagulova, 2020).

Nevertheless, such issues arise when the address of a smart contract may change or may have been communicated incorrectly to end-users. In their practical embodiment, although blockchain itself does promise immutability on deployed contracts, all platform updates or migrations often require that users operate with a new address from another contract. Migrations also give way to a difficult process, disrupted services, and/or financial losses when not taken appropriately by target users to the wrong ones (Bandara et al., 2019). Effective communication strategies include publishing updates through official channels and giving clear instructions to reduce these risks (Hou et al., 2023).

The subject of address assignment also points out a broader challenge of making systems more accessible and usable within the decentralized ecosystem (Zima, 2016). Designers need to think through not only how users locate a contract but also how they would interact with it, in increasingly complex systems that often deal with numerous different contracts and addresses (Nelaturu et al., 2020). Richer user interfaces and tools for the easier management of addresses, like ENS (Ethereum Name Service), could make quite a big difference in facilitating user experience and reducing the rate of user errors (Xia et al., 2022).

TRIGGING AND EXECUTION

By definition, smart contracts are triggered on the occurrence of one or more specific, predefined events, transactions, or conditions encoded in their logic to make them very effective and autonomous. These triggers lie in the code of the contract, acting as a catalyst that executes actions without the use of any intermediary (Bassan & Rabitti, 2024). For example, a user may trigger a smart contract to start a cryptocurrency payment; the smart contract will execute the actions encoded in it, such as transferring ownership of some digital asset, recording a transaction on the blockchain, or sending notifications to parties. In the case of attainment of some date or deadline, the contract could activate automated activities such as the release of funds to beneficiaries, penalization for non-compliance, or the process initiation.

A wide range of triggers for smart contracts demonstrates versatility in certain use cases that are studied upon. For example, a deposit of funds made by users into a certain lending platform triggers the respective smart contract to calculate the interest that is to be returned, disbursement

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

of such is therefore an automated process upon determination, as indicated by (Christidis and Devetsikiotis, 2016). It could also be that, upon delivery, a supply chain actor will have to scan the QR code; in return, through the smart contract, payment will be released at the time of arrival of the goods to their destination (Saberi et al., 2019). Such cases of "if/then" show the efficiency and automation that smart contracts give to many sectors.

These triggers include an inherent logic that leads to actions that are normally executed by smart contracts, by design, to reduce delay or errors associated with manual operation (Shuvo. Mahbub, 2023). For instance, this can be programmed in a way that the smart contract automatically starts interacting every time there is externally sourced data—inputs from an oracle for anything from stock prices or weather data to delivery confirmation. This allows them to respond dynamically to external conditions, expanding their utility in diverse sectors such as supply chain management, insurance, and finance (Beniiche, 2020).

Moreover, the capability of conditional logic programming in smart contracts allows the handling of complex transactional workflows: multistage, multi-agent agreements when certain actions are executed after all conditions predefined are met in order for fairness and compliance to be ensured (Atzei et al., 2017). By nature, the blockchain is immutable and decentralized, adding another layer of security whereby the triggers and outcomes of smart contracts are tamper-proof and thus transparent in the building of trust among parties (Rashid & Siddique, 2019). These features make smart contracts a potentially transformative means of automating processes and raising efficiency and lowering costs in industries.

If some predefined instructions are inculcated into the contract, then the reliability of smart contracts increases manifold. These instructions ensure that the contract functions precisely as intended without ambiguity or external manipulation. However, if unforeseen circumstances arise, then the rigid nature of smart contracts can pose challenges. On the one hand, developers will be able to combine the automation with human oversight involved in a hybrid solution and hence make it more flexible. This way, they still retain blockchain's inherent benefits (De Filippi & Hassan, 2018).

VERIFICATION AND TRANSPARENCY

Verification of the performance of smart contracts is also hugely done through blockchain technology. Once the smart contract is triggered, if the conditions are validated by the blockchain network, then the smart contract will execute its instructions autonomously (Abdelhamid & Hassan, 2019). This whole process initiates when blockchain nodes verify the transaction against the conditions encoded within the contract. All these nodes combined ensure that the conditions are fulfilled before the execution of any event, utilizing decentralization so that tampering or fraud cannot occur (Patel & Singh, 2022). For example, an escrow agreement that has been satisfied will have a smart contract automatically release money to a seller upon confirmation in pre-defined parameters that goods have been received by a buyer, confirmation of delivery, or acknowledgment by the buyer (Asgaonkar & Krishnamachari, 2018).

Decentralized validation from this node-to-node interaction makes the process tamper-resistant and transparent, being recorded immutably, step by step, to the blockchain. This absence reduces associated costs of intermediaries and all potential delays while increasing basic trust among parties (Merlec et al., 2023). In the case of financial services, for instance, processes can be streamlined through smart contracts, thus allowing automatic disbursement where

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

loan conditions—like credit ratings, verified through oracles—meet in real time, given real-time credit standing or income data (Sonawane et al., 2023). Similarly, the smart contract automatically executes supply chain management-related payments when shipment tracking confirms arrival at a destination (Shuvo, 2023). This adds to auditing and compliance due to the transparency of blockchain and its feature of immutability, where each executing step is traceable while having a clear origin in an easily identifiable manner. If a transaction happened because of a smart contract, it is recorded on-chain and, therefore, visibly auditable. That is how trust and accountability are further allowed by all parties involved due to independent verification of the taken acts (Rozario & Vasarhelyi, 2018). Moreover, the cryptographic security of blockchain ensures that once recorded, data cannot be altered or deleted, hence maintaining the integrity of the execution of the contract (Wright & De Filippi, 2015). This transparency goes a long way in helping industries such as supply chain management, where players need to have a view showing the movement and status of goods (Saberi et al., 2019).

Similarly, decentralized execution presents significant benefits, as it removes what is called a "centralized intermediary" in that regard. If there is a smart contract, and this network has been deployed on such a decentralized network, for instance, then realization takes place across many different nodes to ensure redundancy and also to ensure resilience. This, in essence, adds system reliability, with protection from a single point of failure or even tampering. Decentralized performance makes it even more inclusive: agents from different regions are able to interact with the contract without the constraints of centralized oversight (Christidis & Devetsikiotis, 2016). For example, this model is followed by decentralized finance applications to offer users around the world trustless, efficient financial services (Ojog, 2021).

By leveraging blockchain's verification mechanisms, transparency, and decentralized nature, smart contracts offer a robust framework for executing agreements. However, these advantages come with the responsibility of ensuring proper coding, rigorous testing, and adherence to best practices to mitigate potential vulnerabilities and maximize trust in the system (Zhou et al., 2022).

THE LEGAL IMPLICATIONS OF FORMATION PROCESS

The formation process of smart contracts entails a complex interplay of technical and legal considerations. From coding to deployment, address assignment, triggering, and execution, each stage introduces unique challenges that require robust legal frameworks. Addressing issues such as coding errors, liability, privacy, and regulatory compliance is essential to ensure the reliability, fairness, and enforceability of smart contracts in a rapidly evolving digital landscape. On the following will explore the legal implications of the process of forming smart contracts. The discussion that follows will explore the legal implications and complexities associated with each stage of the smart contract formation process, offering insights into how these challenges can be navigated effectively to promote the secure and efficient use of this transformative technology.

CODING CHALLENGES AND LEGAL LIABILITY

The first step of the smart contract creation process is coding and programming, which holds great legal significance. Contractual terms have to be clearly and precisely coded, as even slight ambiguities or errors in coding could lead to disputed interpretations of the code. For instance, a poorly written code might fail to express the intention of the parties to the contract through the creation of gaps that might result in disputes over its

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

interpretation. In fact, such ambiguities let actions unintended by the parties pass through and thus defeat traditional legal frameworks enforcing a contract (Durovic & Janssen, 2019). These problems are further exacerbated by the automated and immutable nature of smart contracts, making the correction of errors post-deployment very challenging and costly (Singh et al., 2023).

Apart from issues related to drafting, legal systems also have to deal with questions of liability in cases where financial or operational losses arise because of vulnerabilities in the code. For example, the critical nature of secure programming practices is shown in such attacks as reentrancy, whereby an attacker abuses functionality to withdraw funds repeatedly (Atzei et al., 2017). In these cases, questions of liability become much more complicated. Should the developer who actually wrote the code be responsible? Or is the liability to lie with the parties approving and deploying the smart contract? These questions, however, make it imperative in the case of any coding errors for legal frameworks to address developer liability and hold someone accountable.

Moreover, shared liability issues arise from the use of pre-audited libraries and third-party components such as OpenZeppelin. While these libraries provide solutions which have been tested and are widely accepted, a set of vulnerabilities in third-party code would cause breaches or losses in several deployed contracts (Huang et al., 2024). When they do, however, an issue of who is responsible arises, whether it be the developer, the library provider, or the parties of the contract, arises and further complicates this already complex legal landscape. This, therefore, calls for the inclusion of indemnity clauses and detailed risk allocation agreements in the development of smart contracts, since liability may arise as pertains to possible third-party dependencies. Such clauses contribute to mitigating uncertainties, since the responsibilities and risks borne by each party are clearly stated, thereby providing a way to resolve disputes.

Another important legal issue concerns intellectual property rights over the smart contract code. Although it would, in theory, be assumed that the code of the smart contract being designed is owned by the developers, in many cases, the level of collaboration among participants leads to situations of licensing and rights usage (Bodó et al., 2018). Meanwhile, when several parties can collaborate in the development or operation of a smart contract—even based on similar code—an ownership and right to usage question needs to be made explicit. Ambiguities in such arrangements may further lead to disputes regarding control, royalties, and liability (Grishchenko et al., 2018). For example, copyright law can be relevant when determining ownership of smart contract code. If a blockchain-based contract is created by multiple developers, the default assumption under copyright law is that each contributor has rights over the code unless explicitly assigned otherwise through a licensing agreement. A dispute may arise if one developer claims exclusive rights and attempts to restrict its use, while others argue for open-source distribution. Similarly, if a smart contract is integrated into a patented blockchain-based process, questions may arise as to whether executing the contract infringes upon an existing patent, especially when the smart contract automates functions covered by the patent's claims. Therefore, ownership, licensing terms, and liability agreements should be clearly defined to avoid legal disputes.

CHALLENGES IN DEPLOYMENT

Further legal issues arise with the deployment of smart contracts. Because of the immutability of blockchain transactions, any errors or omissions in the deployed contract are not easily corrected. This may

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

lead to disputes in cases where deployed code fails to meet the intent of the original agreement, and parties may seek alternative remedies such as deploying new contracts or introducing mechanisms for governance (Zhou et al., 2022). Besides, signing a deploying transaction with a private key means ownership, though there are possible disputes as to who has control over and management of the contract, especially when a project involves collaboration. To this end, the legal system has to consider whether liability lies solely with the deploying party or if other parties have equal liability (Dannen, 2017).

Furthermore, the deployment also has some associated costs, such as gas fees, which may turn contentious if the parties do not agree on its allocation. Thus, high fees or miscalculations may turn into a source of dispute in resource-constrained projects (Chen et al., 2018). The requirement to have proper budgeting and appropriation of deployment costs has to be deliberated within legal frameworks. However, such deployment costs—from among others, gas fees—to be effectively managed, call for clear and concise contractual terms and careful planning with attention to their legal framework and regulatory requirements (Canfora et al., 2020). Parties can reduce contention and build trust by incorporating within the smart contract transparent mechanisms for cost sharing, utilizing technological advances to find cost efficiencies, and ensuring compliance of the arrangement with applicable laws and regulations. Post-deployment reconciliation builds on fairness to ensure that mismatched costs, if any, get equitably settled (Gudmundsson, et al., 2024). A combination of these measures enables a smoother process of deployment in resource-constrained projects and reduces financial as well as legal risks considerably.

BLOCKCHAIN ADDRESS CONCERNS

Assigning unique blockchain addresses to smart contracts presents legal challenges related to accessibility and security, as these addresses are the primary medium for interaction. Errors or fraudulent representations can result in significant financial losses. For example, miscommunication of a contract's address could cause funds to be transferred to unintended destinations, raising critical questions about liability and restitution (Gritti et al., 2023).

Additionally, the issue of migration or change of addresses has its legal consequences, too, and for that, communication protocols need to be strong. If the users are not informed properly about the update, then they may further interact with compromised or obsolete contracts (Garside et al., 2021). The legal systems have to decide whether the liability for such information lies with the originator of the contract or with the platform on which the contract is residing.

To those, add the risks of fraudulent representation of smart contract addresses: villains may actively introduce sham addresses to either misappropriate funds or to mislead users into interacting with spurious contracts. This gives way to a rather complicated legal environment, and in cross-border situations, tracking and recovering misappropriated funds might just be impossible (Madir, 2018). Again, this further complicates the issue of liability when anonymity features inherent in blockchain technologies are utilized to evade detection by perpetrators.

For addressing these concerns, the legal frameworks would have to change and bring in strong mechanisms for accountability and restitution. Platforms and developers may be made to implement checks like address verification systems, checksum algorithms, or multi-signature authentication processes to minimize errors and fraudulent activities (Aitzhan &

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

Svetinovic,2016). In disputes, courts may have to assess the reasonable measures taken by parties and perhaps even fault-based liability or shared responsibility models (Nollkaemper & Jacobs, 2012). Besides that, the regulatory measures would be able to cover increased transparency and consumer education so that users understand the critical importance of handling blockchain addresses with accuracy (Shaji Varughese, 2024). This would help increase confidence in blockchain systems and respond to the peculiar legal challenges introduced by reliance on unique blockchain addresses.

CHALLENGES IN TRIGGERING MECHANISM

Events or conditions that constitute the triggering mechanisms embedded into smart contracts raise some very thorny issues in the realm of jurisprudence, especially regarding consent and intention. These depend on external data supplied by third-party oracles, acting in an intermediary position to feed real-world input into the blockchain. This automation enhances efficiency but equally introduces a number of critical risks. Faulty data or even premeditated manipulation on the part of oracle providers will lead to unintended consequences, such as actual contract execution based on false or manipulated information. In those scenarios, legal liability is problematic to ascertain. There are questions whether the oracle provider owes damages for the defective data or whether the contracting party bears the liability due to a choice of the oracle with an inadequate amount of due diligence (Papadouli & Papakonstantinou, 2023). This gives reason to the importance of the contractual provision for oracle accountability, including arrangements that may reduce disputes by sharing liabilities.

Predefined triggers, along with the logic of execution, also question fairness and adaptability in contract law is that inherent in the rigidity of a smart contract's execution lies the fact that when the conditions are triggered, the contract executes as written, regardless of extraneous variables. The above-mentioned rigidness is opposite to other more traditional legal doctrines such as frustration or impossibility, which allow modification or discharge of contracts if any unexpected event has rendered performance not practicable or unfair. Thus, a natural disaster which causes supply chains to malfunction could bring traditional law relief but may see a smart contract executed and result in losses. Therefore, it creates the need for hybrid frameworks that provide human oversight or a fallback mechanism where manual interference or arbitration in exceptional cases can be facilitated (Molina-Jiménez & Felizia, 2023). It was to this that smart contracts would deal with unexpected events such that whatever solution was decided would remain equitable and without compromising any of the two parties but still provided the advantage of automation.

PRIVACY AND LIABILITY CHALLENGES

While blockchain plays a great role in the verification and execution of smart contracts and provides unparalleled transparency but introduces significant privacy and regulatory concerns. The immutability of blockchain means that contract details and execution history are tamper-proof and auditable, which builds trust among parties. However, this very feature can conflict with data protection laws, such as the General Data Protection Regulation (GDPR), mandating the right to rectify or delete personal data (Tatar et al., 2020). For example, in cases when a smart contract processes personal information, the latter might be unable to exercise his or her rights under these regulations, thus creating some sort of legal tension between the technical design of blockchain and principles of data protection. Such tugged demands might ultimately be reconciled only if innovative solutions—like storing personally identifiable information off-chain but referencing it on the blockchain with cryptographic hashes that can then be changed or deleted without

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

tampering with the blockchain—are adapted by the legal frameworks (Voss, 2021).

Blockchain systems are transparent, an enviable development regarding audit purposes, however, this nature brings serious confidentiality issues to light, particularly in commercial transactions. Public blockchains are designed to permit all participants access to information about smart contracts, a decision that exposes delicate business knowledge such as pricing-sensitive information and proprietary methods. This openness can also dissipate competitive advantages and even lead to the misuse of disclosed information (Sedlmeir et al., 2022). Legal and technological fixes to confidentiality concerns are provided by the use of private or consortium blockchains (Yan Ying et al., 2020). However, these alternatives often come with trade-offs that may undermine exactly those characteristics that give blockchain its value. Indeed, one of the main challenges for the adoption of blockchain in sensitive applications is how to balance confidentiality and decentralization.

Decentralized execution implies more complexity in terms of liability, among other issues. In a distributed system, things happen over a lot of nodes many times without any kind of centralized control. It is in this structure that libility gets dissipated, and thus it is very hard to point out who is responsible in the case of errors, breaches, or even malicious activities (Zetzsche et al., 2017). For example, if a smart contract fails due to some problem in its coding or as a result of certain external manipulations, it will not be quite clear who is responsible—whether the developer, operators of nodes, or users. The legal frameworks will have to fill these lacunas by considering novel approaches, such as attributing liability based on the role played in the network or even decentralized insurance mechanisms to cover losses. Most importantly, industry-wide standards and models of governance will provide critical paths to accountability within the decentralized ethos of blockchain technologies.

CONCLUSION

This paper explores the transformative power of smart contracts in the digital alteration of agreements, emphasizing at the same time major challenges they are facing in the course of their creation and implementation. Smart contracts unleash unparalleled efficiency and reliability in domains like finance, supply chain management, and healthcare since they are self-executing, with inherent transparency supported by immutability. However, many aspects of implementing smart contracts involve complex technical and legal issues that require thorough debate and robust solutions. Every phase in the lifecycle of a smart contract—from coding, deployment, and address assignment to triggering mechanisms and verification—presents unique challenges.

Every step in the smart contract lifecycle—from coding to deployment, address assignment, triggering mechanisms, to verification—introduces unique challenges. Coding errors can lead to vulnerabilities or disputes, while mistakes during on-blockchain deployments are magnified because of their immutable nature. High gas fees and disagreements over their allocation create barriers to adoption. Ownership disputes involve liability and control concerns, especially in a decentralized collaborative environment. The risks include, but are not limited to, mistakes in the communication of blockchain addresses or migrations. Integrating blockchain with external data sources raises legal and privacy concerns, especially under strict data protection laws. Decentralized execution complicates liability and accountability, thus requiring innovative legal solutions.

The way to meet these challenges is through the formulation of appropriate legal

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

frameworks that could give clarity on liability, regulatory compliance, and solutions to mitigate any force majeure events. Besides, mechanisms of governance need to be developed in order to handle erroneous contracts deployed without breaking blockchain's immutability feature. Standards should be defined regarding the cost-sharing agreements so that there are no disputes about the sharing of gas fees and deployment costs. Ownership and control over smart contracts, after their deployment, should be made very clear by the regulators. On the contrary, introduce rigid policies in place that would forcefully notice the users of any address change or migration in order to reduce risks of out-of-date addresses or fraud; encourage the use of user-friendly technologies like ENS, which ultimately will drive uptake and cut errors. Hybrid approaches that implement automation with human oversight, using rich protocols for communication and preserving privacy-enhancing technologies, have shown some promising pathways. Additionally, there will be further development in collaboration between developers, regulators, and industry players that can offer a balanced adaptive ecosystem where smart contracts may be smoothly integrated.

Eventually, the efficiency, cost-cutting, and building of trust in a digitized and decentralized world will be achieved with smart contracts when those challenges are dealt with, and technological innovations align with the legal frameworks. Further research and practical solutions necessary to ensure that the implementation of smart contracts in different applications is reliably, securely, and equitably implemented have been underscored in this study.

ACKNOWLEDGEMENT

No financial aid was received from any person or organization for this article.

CONFLICT OF INTEREST

The authors declare that no conflicts of interest exist.

AUTHORS' CONTRIBUTION

- Conceptualization: All authors
- Data collection: Ra`ed Fawzi Aburoub
- Formal analysis: Ra`ed Fawzi Aburoub & Nabeel Althabahwi
- Funding acquisition: NA
- Investigation; Ra`ed Fawzi Aburoub
- Methodology: Ra`ed Fawzi Aburoub
- Project administration: Nabeel Althabhawi, Mohamad Rizal Abd Rahman & Ammar Abbas Kadhim
- Resources: Ra`ed Fawzi Aburoub
- Software: NA
- Supervision: Nabeel Althabhawi & Mohamad Rizal Abd Rahman
- Validation: NA
- Visualization: NA
- Roles/Writing - original draft: Ra`ed Fawzi Aburoub
- Writing - review & editing: Nabeel Althabhawi & Mohamad Rizal Abd Rahman & Ammar Abbas Kadhim

REFERENCES

Abdelhamid, Manar & Hassan, Ghada. (2019). Blockchain and smart contracts. *In Proceedings of the 8th International Conference on Software and Information Engineering*, 91-95.

Agrawal, R., Chitre, M., & Mahmood, Ahmed. (2016). Design of an address assignment and resolution protocol for underwater networks. *In OCEANS 2016-Shanghai*, 1-7, IEEE.

Ahrendt, W., Pace, G. J., & Schneider, G. (2018). Smart contracts: a killer application for deductive source code verification. Principled Software Development: *Essays Dedicated to Arnd Poetzsch-Heffter on the Occasion of his 60th Birthday*, 1-18.

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

Aitzhan, N. Z., & Svetinovic, D. (2016). Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE transactions on dependable and secure computing, 15(5), 840-852.

Albert, E., Correas, J., Gordillo, P., Román-Díez, G., & Rubio, A. (2020, October). Smart, and also reliable and gas-efficient, contracts. *In 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*,2-2, IEEE.

Andesta, E., Faghih, F., & Fooladgar, M. (2020). Testing smart contracts gets smarter. *In 2020 10th international conference on computer and knowledge engineering (ICCKE)*, 405-412, IEEE.

Angraal, S., Krumholz, H. M., & Schulz, W. L. (2017). Blockchain technology: applications in health care. *Circulation: Cardiovascular quality and outcomes*, 10(9), e003800.

Asgaonkar, A., & Krishnamachari, B. (2019, May). Solving the buyer and seller's dilemma: A dual-deposit escrow smart contract for provably cheat-proof delivery and payment for a digital good without a trusted mediator. In *2019 IEEE international conference on blockchain and cryptocurrency (ICBC)*, 262-267, IEEE.

Atzei. Nicola, Bartoletti. Massimo & Cimoli. Tiziana, (2017). A survey of attacks on ethereum smart contracts (sok). In Principles of Security and Trust: *6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6 Springer Berlin Heidelberg*, 164-186.

Bandara, H. D., Xu, X., & Weber, I. (2020). Patterns for blockchain data migration. *In Proceedings of the European Conference on Pattern Languages of Programs 2020*, 1-19.

Bassan. Fabio & Rabitti. Maddalena, (2024). From smart legal contracts to contracts on blockchain: an empirical investigation. *Computer Law & Security Review*, 55, 106035.

Beniiche, A. (2020). A study of blockchain oracles. *arXiv preprint arXiv*:2004.07140.

Bodó, B., Gervais, D., & Quintais, J. P. (2018). Blockchain and smart contracts: the missing link in copyright licensing? *International Journal of Law and Information Technology*, 26(4), 311-336.

Brent, L., Jurisevic, A., Kong, M., Liu, E., Gauthier, F., Gramoli, V., ... & Scholz, B. (2018). Vandal: A scalable security analysis framework for smart contracts. *arXiv preprint arXiv*:1809.03981.

Buterin. Vitalik, "A next-generation smart contract and decentralized application platform" (2014), *white paper*, 2-1, P 34.

Canfora, G., Di Sorbo, A., Laudanna, S., Vacca, A., & Visaggio, C. A. (2020). Gasmet: Profiling gas leaks in the deployment of solidity smart contracts. *arXiv e-prints, arXiv*-2008.

Chen, T., Li, Z., Zhou, H., Chen, J., Luo, X., Li, X., & Zhang, X. (2018). Towards saving money in using smart contracts. *In Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results,* 81-84.

Christidis. Konstantinos & Devetsikiotis. Michael, "Blockchains and smart contracts for the internet of things" (2016), 4, *Ieee Access,* 2298.

Dannen, C. (2017). Introducing Ethereum and solidity, 1, *Berkeley: Apress,* 159-160

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

De Filippi. Primavera & Hassan. Samer, (2018). Blockchain technology as a regulatory technology: From code is law to law is code. *arXiv preprint arXiv:*1801.02507.

Durovic. Mateja & Janssen. André, (2019). Formation of smart contracts under contract law, The Cambridge handbook of smart contracts, blockchain technology and digital platforms, *Cambridge University Press*, 61-79.

Dwyer. Kevin, (2023). Ethereum's Testnets Explained [2024] Holešky, Goerli, Sepolia, and More, available at: https://www.ankr.com/blog/ethereum-testnets-ultimate-guide/

El Sayed El Gendy. A, (2019), Impact of the use of smart contracts on the efficiency of Islamic banking, Journal of Financial, Accounting & Managerial Studies, Vol 6, No.(2), pp8-12.

Ethereum Foundation. (n.d.), (without year), Ethereum, A Secure Decentralized Generalized Transaction Ledger, Retrieved from, available at https://ethereum.org/

Fischer, M. P., Breitenbücher, U., Képes, K., & Leymann, F. (2017, September). Towards an approach for automatically checking compliance rules in deployment models. *In Proceedings of The Eleventh International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, 150-153.

Fröwis, Michael & Böhme, Rainer. (2017). In code we trust? Measuring the control flow immutability of all smart contracts deployed on Ethereum. In Data Privacy Management, Cryptocurrencies and Blockchain Technology: *ESORICS International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, *Springer International Publishing,* 357-372.

Garside, A., Wilkinson, S., Blycha, N., & Staples, M. (2021). Digital infrastructure integrity protocol for smart and legal contracts diip 2021. Available at SSRN 3814811.

Giancaspro. Mark, "Is a 'smart contract' really a smart idea?" (2017), DOI: 10.1016/j.clsr.2017.05.007. Insights from a legal perspective Article in Computer Law & Security Review. pp1-4.

Godoy, J., Galeotti, J. P., Garbervetsky, D., & Uchitel, S. (2022, October). Predicate abstractions for smart contract validation. *In Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, 289-299.

Grishchenko. Ilya, Maffei. Matteo & Schneidewind. Clara, (2018). A semantic framework for the security analysis of ethereum smart contracts. *In Principles of Security and Trust: 7th International Conference, POST 2018, Held As Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings 7*, *Springer International Publishing,* 243-269.

Gritti, F., Ruaro, N., McLaughlin, R., Bose, P., Das, D., Grishchenko, I., ... & Vigna, G. (2023). Confusum contractum: confused deputy vulnerabilities in ethereum smart contracts. *In 32nd USENIX Security Symposium (USENIX Security 23)*, 1793-1810.

Gudmundsson, J., Hougaard, J. L., & Ko, C. Y. (2024). Sharing sequentially triggered losses: Automated conflict resolution through smart contracts. *Management Science*, 70(3), 1773-1786.

Hasan. Yahya, Legal regulation of electronic contracts, Master thesis,

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

College of Graduate Studies, An-Najah National University, (2007), pp7-8.

Hou, R., Traverson, L., Chabrol, F., Gautier, L., de Araújo Oliveira, S. R., David, P. M., ... & Ridde, V. (2023). Communication and information strategies Implemented by four hospitals in Brazil, Canada, and France to deal with COVID-19 Healthcare-Associated Infections. *Health Systems & Reform,* 9(2), 2223812.

Hu. Yining, Liyanage. Madhusanka, Mansoor. Ahsan, Thilakarathna. Kanchana, Jourjon. Guillaume, Seneviratne. Aruna, (2022). Blockchain-based smart contracts-applications and challenges, https://doi.org/10.48550/arXiv.1810.04699 , PP1-26.

Huang, M., Chen, J., Jiang, Z., & Zheng, Z. (2024, February). Revealing Hidden Threats: An Empirical Study of Library Misuse in Smart Contracts, *In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering* 1-12.

Hwang. Seon-Jin, Choi. Seok-Hwan, Shin. Jinmyeong, Choi. Yoon-Ho, (2022). CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection, *IEEE Access*, 10, 32595.

Imteaj. Ahmed, Amini. M. Hadi, Pardalos. Panos M, (2021). Toward smart contract and consensus mechanisms of Blockchain. Foundations of Blockchain: Theory and Applications, ISBN : 978-3-030-75024-4, 15-28.

Khan, N., Lahmadi, A., Francois, J., & State, R. (2018, April). Towards a management plane for smart contracts: Ethereum case study. *In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 1-6, IEEE.

Knecht, M., & Stiller, B. (2022). CASC: Content Addressed Smart Contracts. *In 2022 IEEE International Conference on Blockchain (Blockchain)*, 326-333, IEEE.

Kuppuswamy, S., & Kodavali, L. (2024). Chapter Six-Bayesian network-based quality assessment of blockchain smart contracts. *Adv. Comput*, 132, 85-110.

Loddo, O. G., Addis, A., & Lorini, G. (2022), Intersemiotic translation of contracts into digital environments, Frontiers in Artificial Intelligence, pp5-8.

Madir, J. (2018). Smart contracts:(how) do they fit under existing legal frameworks?. Available at SSRN 3301463.

Merlec, M. M., Sinai, N. K., & In, H. P. (2023, October). A Blockchain-based Trustworthy and Secure Review System for Decentralized e-Portfolio Platforms. *In 2023 14th International Conference on Information and Communication Technology Convergence (ICTC),* 675-680, IEEE.

Milosevic, Z., Josang, A., Dimitrakos, T., & Patton, M. A. (2002), Discretionary Enforcement of Electronic Contracts, Central Laboratory of the Research Councils, pp8-11.

Molina-Jimenez, C., & Felizia, S. M. (2024). On the use of smart hybrid contracts to provide flexibility in algorithmic governance, *Data & Policy*, 6, e8.

Nelaturu, K., Mavridou, A., Stachtiari, E., Veneris, A., & Laszka, A. (2022). Correct-by-design interacting smart contracts and a systematic approach for verifying ERC20 and ERC721 contracts with VeriSolid. *IEEE Transactions on Dependable and Secure Computing*, 20(4), 3110-3127.

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

Nelaturu, K., Mavridoul, A., Veneris, A., & Laszka, A. (2020, May). Verified development and deployment of multiple interacting smart contracts with VeriSolid. *In 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1-9, IEEE.

Nollkaemper, A., & Jacobs, D. (2012). Shared responsibility in international law: a conceptual framework, *Mich. J. Int'l L.,* 34, 359.

Nugraheni, N., Mentari, N., & Shafira, B. (2022), The Study of Smart Contract in the Hara Platform under the Law of Contract in Indonesia, DOI: 10.36348/sijlcj.2022.v05i07.005, ISSN 2616-7956 (Print) |ISSN 2617-3484 (Online), Scholars International Journal of Law, Crime and Justice, pp277-278.

Ojog, S. (2021). The emerging world of decentralized finance. *Informatica Economica*, *25*(4), 43-52.

Oliva, G. A., & Hassan, A. E. (2021, August). The gas triangle and its challenges to the development of blockchain-powered applications. *In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1463-1466.

Oluwatosin. Serah, (2023). Mastering Addresses In Ethereum, avilabile at: https://medium.com/@ajaotosinserah/mastering-addresses-in-ethereum-5411ba6c3b0f

Otieno, M., Odera, D., & Ounza, J. E. (2023). Theory and practice in secure software development lifecycle: A comprehensive survey, *World Journal of Advanced Research and Reviews*, 18(3), 053-078.

Pacheco, M., Oliva, G. A., Rajbahadur, G. K., & Hassan, A. E. (2023). What makes Ethereum blockchain transactions be processed fast or slow? An empirical study. *Empirical Software Engineering*, 28(2), 39.

Papadouli, V., & Papakonstantinou, V. (2023). A preliminary study on artificial intelligence oracles and smart contracts: A legal approach to the interaction of two novel technological breakthroughs. *Computer Law & Security Review*, 51, 105869.

Patel, A., & Singh, B. (2022). Implementation of Smart Contract Using Ethereum Blockchain. *In International Conference on Advanced Communication and Intelligent Systems*, *Cham: Springer Nature Switzerland,* 160-169.

Pereira. Carla Roberta, da Silva. Andrea Lago, Tate. Wendy Lea, Christopher. Martin. (2020). Purchasing and supply management (PSM) contribution to supply-side resilience, *International Journal of Production Economics*, 228(6),107740.

Qutieshat, E., Al-Tarawneh, B., & Al Naimat, O. (2022), The Legal Status of Smart Contracts According to the Jordanian Civil Law Theory of Contracts. Volume 14. No. 4, Jordanian Journal of Law and Political Science, pp12-14.

Rashid, A., & Siddique, M. J. (2019, February). Smart contracts integration between blockchain and internet of things: Opportunities and challenges. *In 2019 2nd International Conference on Advancements in Computational Sciences (ICACS)*, 1-9, IEEE.

Restack, (2024). Smart Contract Vulnerabilities Scanning Tools, available at: https://www.restack.io/p/smart-contract-vulnerabilities-answer-scanning-tools

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

Rozario, A. M., & Vasarhelyi, M. A. (2018). Auditing with Smart Contracts, *International Journal of Digital Accounting Research*, 18.

Saberi, S., Kouhizadeh, M., Sarkis, J., & Shen, L. (2019). Blockchain technology and its relationships to sustainable supply chain management. *International journal of production research*, 57(7), 2117-2135.

Savelyev. Alexander, (2017). Contract law 2.0: 'Smart' contracts as the beginning of the end of classic contract law, *Information & Communications Technology Law,* DOI: 10.1080/13600834.2017.1301036, 126.

Sedlmeir, J., Lautenschlager, J., Fridgen, G., & Urbach, N. (2022). The transparency challenge of blockchain in organizations. *Electronic Markets*, *32*(3), 1779-1794.

Seitenov, A., & Smagulova, G. (2020). Distribution of ethereum blockchain addresses. Scientific Journal of Astana IT University, (4), 41-48.

Semnani, A., & Yang, G. (2024). Beyond Collectibles: A Comprehensive Review of Non-Fungible Token Applications. Available at SSRN 5046792.

Seneviratne, O. (2024). The Feasibility of a Smart Contract" Kill Switch". *arXiv preprint arXiv*, 2407.10302.

Shaji Varughese, J. (2024). Streamlining Regulatory Processes with Blockchain Technology: Case Studies and Best Practices, *International Journal of Science and Research* (IJSR), 1-6.

Sharma, T. (2020). Installing Frameworks, Deploying, and Testing Smart Contracts in Ethereum Platform, In Bitcoin and Blockchain, *CRC Press* 137-162.

Shein, Esther. (2021). Converting laws to programs, *Communications of the ACM*, 65(1), 15-16.

Shuvo. Mahbub, (2023). Smart Contracts in Supply Chain – 4 Challenges of Implementing, available at: https://coredevsltd.com/articles/smart-contracts-in-supply-chain/

Signer, C. (2018). Gas cost analysis for ethereum smart contracts, (Master's thesis, *ETH Zurich, Department of Computer Science).* 20-35.

Sindhavad. Aditya, Yadav. Ramavtar & Borse. Yogita, (2023). Crowdfunding using Blockchain. *In 2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 1-6, IEEE.

Singh, J., Sahu, D. P., Murkute, S., Yadav, U., Agarwal, M., & Kumar, P. (2023). Deep Learning Based Bug Detection in Solidity Smart Contracts. *In International Conference on Recent Trends in Image Processing and Pattern Recognition, Cham: Springer Nature Switzerland,* 101-109

Sonawane, N., Gupta, P., Laksh, C., & Gururaja, H. S. (2023, October). A Blockchain Solution for Enhancing Risk Management and Transparency in Loan Disbursements. *In 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT),* 1-6, IEEE.

Stampernas. Sotirios, (2018), Blockchain technologies and smart contracts in the context of the Internet of Things, Master thesis. University of Piraeus, pp26-28.

Szabo. Nick, (1996). Smart Contracts: Building Blocks for Digital Markets, Extropy, #16. Original publication. available online: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool200

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]

6/szabo.best.vwh.net/smart_contracts_2.html

Tatar, U., Gokce, Y., & Nussbaum, B. (2020). Law versus technology: Blockchain, GDPR, and tough tradeoffs. *Computer Law & Security Review*, 38, 105454.

Timuçin, Tunahan & Biroğul, Serdar. (2023). THE EVOLUTION OF SMART CONTRACT PLATFORMS: A LOOK AT CURRENT TRENDS AND FUTURE DIRECTIONS. *Mugla Journal of Science and Technology*, 9(2), 46-55.

Tok. E & Tunca. C, "Is Every Electronic Contract A Smart Contract" (2023), available at https://rb.gy/f30rut,visited on 19/03/2025 at 9:35pm.

Vasiu. Ioana & Vasiu. Lucian, (2023). A Framework for Effective Smart Contracting, *Bratislava Law Review*, 7(2), 107-122.

Voss, W. G. (2021). Data protection issues for smart contracts. Smart Contracts: Technological, Business and Legal Perspectives (Marcelo Corrales, *Mark Fenwick & Stefan Wrbka, eds., Hart Publishing/Bloomsbury, 2021),* https://www.bloomsburycollections.com/book/smart-contracts-technological-business-and-legal-perspectives .

Wood, G. (2016). Polkadot: Vision for a heterogeneous multi-chain framework. *White paper*, 21(2327), 4662.

Wood, Gavin. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.

Worley. Carl & Skjellum. Anthony, (2019). Opportunities, challenges, and future extensions for smart-contract design patterns. *In Business Information Systems Workshops: BIS 2018 International Workshops, Berlin, Germany, Revised Papers 21*, *Springer International Publishing,* 264-276.

Wright. Aaron & De Filippi. Primavera, (2015). Decentralized blockchain technology and the rise of lex cryptographia. Available at SSRN 2580664.

Xia, P., Wang, H., Yu, Z., Liu, X., Luo, X., Xu, G., & Tyson, G. (2022, October). Challenges in decentralized name management: the case of ENS. *In Proceedings of the 22nd ACM Internet Measurement Conference,* 65-82.

Yan, Y., Wei, C., Guo, X., Lu, X., Zheng, X., Liu, Q., ... & Jiang, G. (2020, June). Confidentiality support over financial grade consortium blockchain. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, 2227-2240.

Zarir, A. A., Oliva, G. A., Jiang, Z. M., & Hassan, A. E. (2021). Developing cost-effective blockchain-powered applications: A case study of the gas usage of smart contract transactions in the ethereum blockchain platform. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(3), 1-38.

Zetzsche, D. A., Buckley, R. P., & Arner, D. W. (2018). The distributed liability of distributed ledgers: Legal risks of blockchain. *U. Ill. L. Rev.,* 1361.

Zhou. Haozhe, Milani Fard. Amin & Makanju. Adetokunbo, (2022). The state of ethereum smart contracts security: Vulnerabilities, countermeasures, and tool support. *Journal of Cybersecurity and Privacy*, 2(2), 358-378.

Zima, M. (2016). Sending Money Like Sending E-mails: Cryptoaddresses, Universal Decentralised Identities. *arXiv preprint arXiv*, 1612.04982.

[Received: 15 January 2025; Accepted: 17 April 2025; Published: xx xx xxxx]