International Journal of Information Technology and Web Engineering
Volume 17 • Issue 1

Smart Contracts Security Threats and Solutions

Senou Mahugnon Rosaire, Institute of Mathematics and Physics, University of Abomey-Calavi, Benin*
Degila Jules, Institute of Mathematics and Physics, University of Abomey-Calavi, Benin
https://orcid.org/0000-0003-4688-9178

ABSTRACT

Blockchain-enabled smart contracts are subjected to several issues leading to vigorous attacks such as the decentralized autonomous organization (DAO) and the ParitySig bug on the Ethereum platform with disastrous consequences. Several solutions have been proposed. However, new threats are identified as technology evolves and new solutions are produced, while some older threats remain unsolved. Thus, the need to fill the gap with a more comprehensive survey on existing issues and solutions for researchers and practitioners arises. The resulting updated database will become an essential means for choosing a particular solution for a specific subject. In this review, the authors embrace mainly codifying security privacy and performance issues and their respective solutions. Each problem is attached to its corresponding solutions when they exist. A summary of the threats and solutions is provided as well as the relationship between threat importance and the given answers. They finally enumerate some directives for future works.

KEYWORDS

Blockchain, Codifying, Concerns, Intelligent Agreement, Performance, Privacy, Safeness, Solving

INTRODUCTION

Transactions are subject to risks and high transaction fees in the real world, especially when third parties are involved. The creation of blockchain technology brought the light of hope to this issue. The Blockchain is a decentralized ledger that allows secure transactions at a low cost (Nakamoto, 2008). At first, its application was related to the finance domain using the Bitcoin currency but has been extended lately to several sectors, including the contract domain, giving rise to a new intelligent contract era. Smart contracts based on Blockchain are a set of codes that enforce contract execution. Their correctness is essential to ensure trust in blockchain-based systems (Alexandre et al., 2018).

The contracts are presented as programs running on blockchain platforms, such as the Ethereum Virtual Machine (EVM). They interact through well-defined interfaces where no third parties are involved in the financial transaction's completion in a distributed environment. However, attackers can

DOI: 10.4018/IJITWE.304048 *Corresponding Author

exploit security vulnerabilities from the interfaces as they provide favorable malicious deeds (Yu et al., 2019). Blockchain technology has an immutability property that does not simplify bug fixing in a smart contract. One of the main reasons is that a deployed smart contract is not modified directly when bugs are found. It involves assets and different parties that need to be considered carefully. A novel version of the contract is created and deployed to fix bugs. However, data on the previous contract is not automatically sent to the new contract. Manual intervention is required to initiate the new contract with the earlier data, which is very clumsy (Shuai et al., 2019). Several smart contract vulnerabilities have been noticed; a breach of trust in the underlying blockchain technology was revealed (Evgeniy, 2018). Two of the most notorious security breaches are the infamous Decentralized Autonomous Organization (DAO) exploit, which has led to a considerable loss of more than $50 million, and the ParitySig attack, where $169 million is locked forever (Yuepeng et al., 2019; Franklin et al., 2019).

These incidents shed more light on the importance of securing smart contracts, and the user community started to pay more attention to them. Therefore, programmers are forced to ensure that smart contract codes are challenged from security perspectives before deployment. Consequently, smart contract issues have been hot topics among researchers (Wang et al., 2018). The frequent threats in the literature are related to codifying, security privacy and performance aspects, smart contract life cycle, and the blockchain architecture layers (Maher & Aad Van, 2017; Zibin et al., 2019; Huashan et al., 2020). Several contract analysis tools have been developed in the past few years to address these concerns and consist of an important database that needs to be continuously updated. Indeed, such a document is a powerful tool to help and guide blockchain practitioners and researchers. As most difficult issues are tackled, it is a severe option to ensure minimum security on the blockchain smart contract. However, threats have been discovered. Previous tools become obsolete against new threats, and security is therefore not assured. This situation becomes a concern and would lead Blockchain and smart contract users to conduct dense and thorough research to ensure smart contract safety. Also, as some issues are yet to be tackled, and some have received more attention than others, it becomes relevant to bring to their knowledge a clear vision of smart contract security for a better tool choice.

Thus, we provide this review to combine several smart contract problems with their respective solutions as much as possible. This article surveys the literature of smart contract issues from 2014 to 2021 as they apply to codify, security, privacy, and performance domain. Solutions related to issues are classified accordingly, while suggestions for likely research directions are presented.

Our contributions from this work are described as follows:

Identifying new vulnerabilities concerning codifying, security, privacy, and performance issues.

Identifying newly developed solutions against vulnerabilities.

Creating a new database that can serve as a guideline in the smart contract security domain with insights on future directions to help researchers, students, and practitioners.

Section 2 talks about the problem statement and objectives in the rest of the paper. Section 3 presents Blockchain and smart contracts. Then, section 4 tackles the research methodology, while section 5 shows the results obtained. After that, the discussion of the results is conducted in section 6, and section 7 presents research perspectives. Finally, section 8 concludes the paper.

PROBLEM STATEMENT AND OBJECTIVES

Several blockchain-enabled smart contract issues have been noticed, and solutions have been provided accordingly. Researchers combined the solutions to produce a literature review document related to the topic. However, the rapid growth of threats and solutions makes the previous database obsolete. Indeed, when referencing the existing database, only old bugs and related solutions are found. This results in a waste of time, leading to conducting more studies. As of this writing, as far as we know, only one database related to codifying, security, privacy, and performance was created in 2017, which is indeed outdated due to the fast development of the technology.

Smart contracts are involved in several domains where a considerable amount of money must be secured as much as possible. Indeed, using the wrong tool to verify blockchain security thoroughly would result in security breaches, leading to loss of money.

This work's contribution is to draw smart contracts users' attention to the risks related to using this technology and how best they can secure or check their smart contract for a particular issue with the most appropriate tool while setting a new referencing database.

This includes:

Providing a current database that groups codifying, security, privacy, and performance issues with the most appropriate tool for their detection. This brings into combined document methods or new methods to detect threats.

Providing guidelines regarding the most recurrent and dangerous issues. Indeed, the more notorious a threat is, the more drastic measures are needed to counter it. This helps smart contract practitioners grasp how far research has evolved to secure smart contracts while allowing them to instantly. choose from among various tools.

Proposing future research perspectives.

Setting a new benchmark for the issues involved in the study. Though research has evolved, greater improvements are necessary to secure smart contracts better.

THE STATE OF THE ART

Blockchain and Smart Contracts: A Brief Overview

A blockchain is a distributed system that allows transaction completion without a third party (Nakamoto, 2008). The blockchain system consists of blocks with several transactions connected to a precedent block to form a chain. There is a genesis block to which the other blocks are appended, and every generated block goes through a validation process (Zibin & al., 2019) before it joins the chain. The reliability of the chain comes from the consensus algorithms. Some of them, such as Proof of Work (PoW), Proof of State (PoS), and Practical Byzantine-Fault Tolerance (PBFT), are used to solve a puzzle with the help of miners for block validation among nodes. The miner is responsible for selecting new transactions, recording them in the new block, and executing those contracts (Franklin & al., 2019). Solving the puzzle is difficult as the process must at the same time secure the Blockchain by preventing attackers from forging it. As a result, Blockchain exhibits decentralization, integrity, and auditability (Shuai et al., 2019), which allows its adoption.

Blockchain-based smart contracts are systems that get executed on distributed nodes without a central authority, allowing companies to collaborate easily while enforcing contract clauses (Chibuzor et al., 2018). Each smart contract receives a unique address, and for its execution, a transaction is initiated between a sender and the smart contract. A computational cost is needed to complete the transaction. The unit of the transaction cost is gas. The used gas gives the block miner where the transaction is kept, and the sender returns the unused gas. A threshold value for gas usage is specified to prevent unnecessary gas usage due to unoptimized programs. An exception is thrown when the gas threshold value is attained (Daniel & Benjamin., 2019). The contract execution follows a predefined process involving all network nodes concerning the triggering transaction data. According to (Konstantinos & Michael., 2016), each node involved in the smart contract blockchain-based runs a virtual machine (VM).

As the blockchain network consists of several nodes, it is then considered to be a distributed VM. The smart contract source code is available, and once deployed, it cannot be changed. Therefore, the executed code must be error-free while ensuring trust and fulfillment of intended use (Franklin & al., 2019). Smart contracts are created with solidity, one of the programming languages designed for it. It is mainly used by programmers and is similar to an object-oriented language (Zeinab & al., 2018).

RELATED WORK

Several studies have been conducted on blockchain-based smart contracts. Authors (Chibuzor & al., 2018) produced a systematic review of older studies emphasizing smart contracts application in organizations. The implementation of smart contracts for the internet of things (IoT) has been tackled by (Konstantinos & Michael., 2016) in their work, where combining the two technologies is a significant concern. Evgeniy (2018) brought upfront smart contract security concerns while providing a symbolic model checking that ensures smart contract business logic correctness. Other solutions such as Oyente, Mythril, Madmax, Vandal et ContractFuzzer, security, SmartCheck, Zeus, etc., are proposed for vulnerabilities, and are summarized in Reza et al. (2018), Sukrit et al. (2018), and Daniel and Benjamin (2019). Greedy, Prodigal, and Suicidal contracts categorize attacks that the authors identified (Ioannis et al., 2018). They implemented Maian, a symbolic execution tool for detecting such attacks. These attacks are also tackled by Wesley et al. (2019) using Long Short Term Memory (LSTM), a machine learning technique. This machining technique has been further improved in (Peng & al., 2020) work with bidirectional LSTM (BLSTM) and BLSTM based on an attention mechanism for detecting reentrancy attacks on blockchain-enabled smart contracts.

A systematic mapping study on smart contracts was conducted Maher and Aad Van (2017). This study focused on smart contracts' key parameters, codifying issues, and other smart contract-related topics, including smart contract application topics. Zibin et al. (2019) provide a study on smart contracts while addressing challenges during the different phases (creation, deployment, execution, completion) of their existence. Huashan et al. (2020) tackle ethereum blockchain attacks, vulnerabilities, and defenses, and Jorge et al. (2019) provide a review on blockchain privacy issues. Monika & Gernot (2019) produced a document that embraces smart contract tools for vulnerability detection is produced. Sarwaar et al. (2020) describe the most (10) tools for threat detection while emphasizing their limitations.

Suvitha & Subha (2021) explored various smart contract platforms and their features for appropriate use. Negara et al. (2021) conducted a literature review with an exploratory approach where frameworks, methods, and simulations of smart contract implementations in various domains are reviewed. In their survey, Garfatta et al. (2021) presented a general overview of smart contracts verification, mainly formal verification. They provided results related to formal method approaches as well as threats they can tackle. Samreen and Alalfi (2021) conducted a survey where eight vulnerabilities and their detection tools were reviewed. Khan et al. (2021) categorized the existing blockchains' interoperability solutions into three main categories: heterogeneous blockchains and homogeneous smart contracts, homogeneous blockchains and homogeneous smart contracts, heterogeneous blockchains, and heterogeneous smart contracts. They showed how to fill the gap between these solutions using smart contracts and provided further research orientations.

Wang et al. (2021) provide a smart contract review of the current research status and advances into six categories: symbolic execution, abstract interpretation, fuzz testing, formal verification, deep learning, and privacy enhancement. Tools comparison and methods developed for solving threats are also provided. Timuçin and Birogul (2021) conducted a study related to the transformation of smart contracts into real "smart" contract structures with the use of Deep Learning algorithms, while Khan et al. (2021) present a comprehensive survey about blockchain-based smart contracts from technical and usage points of view. Peng et al. (2021) performed a comprehensive review of Smart contract applications in the IoT domain for security threats. They also provided directions for further research.

Tang et al. (2021) analyze 15 security vulnerabilities and provide causes and methods used to address them. They also analyzed existing solutions, methods, and machine learning techniques, and advised using the two strategies to address security issues better and detect new threats.

METHODOLOGY

We systematically review smart contracts challenges and solutions in particular domains such as codifying, security, privacy, and performance, while emphasizing the most tackled issues. Our methodology involves five steps: research question identification, searching methods and paper downloading, paper screening, paper analysis, and review writing (Maher & Aad Van., 2017).

In the research question identification, we specified the questions that the study would attempt to tackle: What is the current state of blockchain-based smart contract issues and solutions approaches? What vulnerabilities do smart contracts suffer, especially in the codifying, securities, privacy, and performance sectors? Is every vulnerability solved? What are the tools used to tackle those issues? What is the relationship between vulnerabilities and solutions? What could future trends be?

We proposed a search string using appropriate words such as smart contract problems or challenges or threats or bugs or vulnerabilities and approaches or detection techniques or solutions. This is to bring down all papers that could be related to our study.

We then explored several databases that are mainly used for academic research, such as Google Scholar, Springer, IEEE, ACM, and Preprint arXiv, to collect conference papers and journal articles. However, as academic literature can lack newly disclosed events or materials, we also used Google to look for reports, white papers, documentation, and websites that could provide relevant information. Through this exploration, several papers were downloaded and categorized according to the issues they tackled.

The next stage involved paper screening. We removed papers that did not satisfy specific criteria for adequate analysis. As criteria, we considered papers without full text, smart contracts papers concerning technologies other than Blockchain, redundant/duplicate articles, and documents with non-important abstracts. Articles that tackle smart contract concerns related to our objectives were kept, along with essential reports, documentation, and websites.

Next, relevant papers were obtained. They were analyzed, and all information pertinent to our objectives was extracted. Indeed, threats that fall within one of these categories of vulnerabilities--codifying, security, privacy, and performance—and their solutions, were considered and grouped while future trends were also provided from the relevant data gathered.

Finally, we combined the different information to write the review.

RESULTS

We produced the following tables from the papers analyzed, which consider smart contract issues with the developed solutions.

Figure 2 shows the relationship between the different types of issues together with their solutions.

Security issues have been of significant concern as we denote 18 different cases derived from it, leading to more than 130 solutions. They receive much attention far beyond the codifying, privacy, and performance issues with 4, 2, and 2 problem cases identified with 66, 40, and 34 solutions developed, respectively.

CODIFYING ISSUES

Figure 3 depicts the number of provided solutions against codifying issues.

Issue 1 = Under-optimized smart contract identification
Issue 2 = Correct smart contract programming language
Issue 3 = Complexity of programming language
Issue 4 = Smart Contract termination or modification weaknesses

Figure 1. Research Process (Adapted from Maher and Aad Van, 2017)

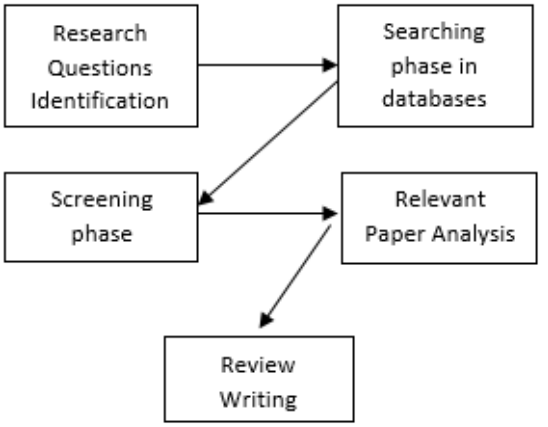


Table 1. Summary of Identified Problems and Solutions

Type of category	Number of issues per category	Number of solutions per category
Codifying	4	66
Security	18	132
Privacy	2	40
Performance	2	34

Table 2. Smart Contracts Problems With Developed Solutions in Detail

Smart Contract Problems		Techniques/Developed Tools
Codifying issues	Trouble with correct writing of smart contracts	Adoption of formal verification methods (Maher & Aad Van, 2017; Jing & Zhentian, 2019; Tesnim & Kei-Leo, 2018). Annotary (Konrad and Julian., 2019). Guidelines/standards for developers (Kevin et al., 2016; Bartoletti & Pompianu, 2018). ContractWar (W. Wang et al., 2020). ContractGuard (X. Wang et al., 2020). SmartDEAMP (Zibin & al., 2019). Gigahorse (Zibin & al., 2019). Semi automation of smart contract creation (Christopher & Mariusz ., 2016). New contract language development (Jing & Zhentian., 2019). Semantics analysis (Zibin et al., 2019; Anastasia & Aaron, 2018). Raziel (David, 2020). Solidifier (Antonino & Roscoe., 2020). Verismart (Sunbeom & al., 2019). VeriSolid (Anastasia & Aron., 2019). Osiris (Christof & al., 2018). SASC (Zhou et al., 2018). SCRepair (Xiao et al., 2020). SmartShield (Yuyao et al., 2020). Manticore (Mark & al., 2019). FSolidM (Anastasia & Aaron, 2018). VerX (Anton et al., 2020). Zeus (Sukrit et al., 2018). iContract (Qasse & al., 20201). Smart-Graph (Pierro., 2021). SuMo (Barboni & al., 2021). A security verifier type (Hu & al., 2021).
	Smart Contracts termination or modification weaknesses	Use of norms for smart contracts (Maher & Aad Van., 2017; Bill & Ari., 2016). Mechanized proof of termination of smart contract (Thomas et al., 2020). Proof-carrying smart contracts (Hu & al., 2021). *Intelligible Description Language Contract (IDLC)*,
	Under-optimized smart contracts identification	SmartCheck (Sergei & al., 2018). MadMax (Neville & al., 2018). Maian (Ivica N. & al., 2018). Securif (Petar & al., 2018). SmartCopy (Yu & al., 2019). Gas reducer (Ting & al., 2018). Gastap (Sara & Ralph., 2019). Gasper (Bill and Ari., 2016). Gasol (Elvira & al., 2019). GasChecker (Ting et al., 2020). GasFuzzer (Ashraf et al., 2020). SafeVM (Elvira & al., 2019). SolAnalyser (Akca & al., 2019). Mythril (Sarwaar et al., 2020). Contract version comparisons, Graph Neural Networks (Yuan & al., 2020). sCompile (Jialiang et al., 2019). SCRepair (Yu et al., 2020). Formal verification framework (FVF) (Maher & Aad Van., 2017; Jing & Zhentian, 2019). SmartShield (Yuyao et al., 2020). Manticore (Mark & al., 2019). Zeus (Sukrit et al., 2018). Echidna (Gustavo & al., 2020). Eth2vec (Ashizawa & al., 2021). Machine Learning Approach for Gas Price Prediction in Ethereum Blockchain (Mars & al., 2021).
	Complexity of programming languages	Logic-based languages (Prolog) (Zibin et al., 2019; Florian et al., 2016). IELE (Scillia, Yul) (Tyurin & al., 2019). Use of type based-language Idris, Simplicity (O'Connor R., 2017). Liquidity (Çagdas & al., 2018). Obsidian (Coblenz, 2017). Flint (Schrans & al., 2018). Mandala (Markus, 2019). SmaCoNat (Regnath & Steinhors ., 2018). Bitml (Tyurin & al., 2019). SPESC (Xiao & al., 2018).

continued on following page

Table 2. Continued

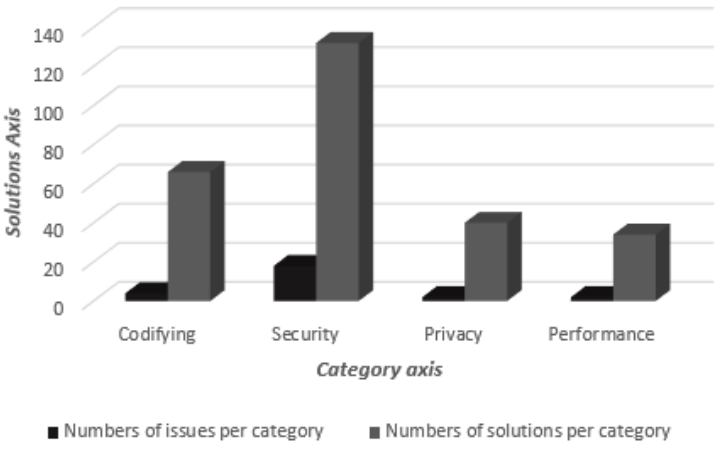
Smart Contract Problems		Techniques/Developed Tools
Securities issues	Transaction-ordering vulnerability	Ethereum Based Functions (use of SendIfReceived function) (Christopher & Vincent., 2016). Oyente (Loi & al., 2016). Ether racer (Aashish & al., 2019). Fuzzer symbolic execution (Jingxuan & al., 2019). FSolidM (Anastasia & Aaron, 2018). ContractWar (Wang Wei & al., 2020). Zeus (Sukrit et al., 2018). Securify (Petar & al., 2018). Mythril (Sarwaar et al., 2020). Unsecured smart contract detection (Kevin & al., 2021).
	Timestamp vulnerability	Random seed of block number (Loi & al., 2016). SIF (Chao & al., 2019). Slither (Josselin & al., 2019). SmartCopy (Yu & al., 2019). ContractWar (Wang et al., 2020). Mythril (Sarwaar et al., 2020). Graph Neural Networks (Yuan & al., 2020). Fuzzer symbolic execution (Jingxuan & al., 2019). SolAnalyser (Akca & al., 2019). Eth2vec (Ashizawa & al., 2021). SmartPulse (Stephens & al., 2021). Graph Neural Network with expert knowledge (Liu & al., 2021).
	Mishandled exception vulnerability	Check of the return value (Loi & al., 2016). ContractFuzzer (Jiang & al., 2018). Vandal (Brent & al., 2018). Zeus (Sukrit et al., 2018). Contract version comparison, SmartCheck (Sergei & al., 2018). Securify (Petar & al., 2018). SmartCopy (Yu & al., 2019). Mythril (Sarwaar et al., 2020). ContractWar (Wang Wei & al., 2020). Slither (Josselin & al., 2019). Oyente (Loi & al., 2016). Fuzzer symbolic execution (Jingxuan & al., 2019). EVMPatch (Rodler & al., 2020). SolAnalyser (Akca & al., 2019). Elysium (Torres & al., 2021).
	Block randomness	Delay and Sloth function (Zibin & al., 2019).
	BGP Routing concerns	Sabre (Maria A. & al., 2017).
	Reentrancy vulnerability	ÆGIS (Torres & al., 2020). ReGuard (Chao & al., 2018). Reentrancy analyzer (Chinen & al., 2020), Bidirectional LSTM – ATTention (BLSTM-ATT) (Peng & al., 2020). ECFChecker (Grossman & al., 2017). EthScope (Wu & al., 2020). Sereum (Michael & al., 2019). Teether (Johannes and Christian., 2018). Zeus (Sukrit et al., 2018). Graph Neural Networks (Yuan & al., 2020). SmartCheck (Sergei et al., 2018). Securify (Petar & al., 2018). SmartCopy (Yu & al., 2019). Manticore (Mark & al., 2019). Mythril (Sarwaar et al., 2020). Oyente (Loi & al., 2016). ContractFuzzer (Jiang & al., 2018). ContractWar (Wang et al., 2020). TxSpector (Mengya & al., 2020). Vandal (Brent & al., 2018). Slither (Josselin & al., 2019). Mechanism to Detect and Prevent Ethereum Blockchain Smart Contract (Alkhalifah & al., 2021). Dynamit (Eshghie & al., 2021). Elysium (Torres & al., 2021). Eth2vec (Ashizawa & al., 2021). Reentrancy detection using TXL programming language (Samreen & Alalfi., 2020). Graph Neural Network with expert knowledge (Liu & al., 2021). EtherSolve (Contro & al., 2021). SGuard (Nguyen & al., 2021). EtherClue (Aquilina & al., 2021). VSCL (Mi & al., 2021).
	Ponzi scheme issues	Machine learning techniques (Ripper, Bayes Network and Random Forest, XGBoost) (Weili & al., 2018). Al-SPSD: Anti-leakage smart Ponzi schemes detection in Blockchain (Fan & al., 2021).
	HoneyPot	HoneyBadger (Christof & Mathis., 2019).
	Delegated Puzzle	Piece of work (Philip & al., 2017). Non-delegated Scratch-Off Puzzles (Miller & al., 2015). Sign to Mine (Ziftr., 2014). Phase-Proof of Work (2P-PoW) (Eyal & Sirer., 2014).
	Man in the middle	Role-Based Access Control (RBAC) (Kamboj & al., 2021).
	Denial of Service with Block Stuffing	Smart Contract-Based Solution for Secure Distributed SDN (Almakhour & al., 2021). SmartScan (Samreen & Alalfi., 2021). Co-Chain SC (Houda & al., 2019).
	Selfish Mining approached	Computation power reduction (to ¼), still an open research area approached (Cyril & Ricardo., 2019; Ittay & Emin, 2018). TFCrowd (Li & al., 2021).
	Verifier's dilemma	Consensus computational framework (Loi & al., 2015).
	Criminal/Opaque smart contract activities	Trustee Neutralizable smart contract (Ari & al., 2016). Erays (Yi & al., 2018). Smart Inspect (Santiago et al., 2018). Samos (Knecht & Stiller., 2021). Smart contract-based Supply Chain Control (Dietrich & al., 2020).
	Untrustworthy data feeds issues	Town Crier (TC) (Fan & al., 2016). Reputation contract (Zibin & al., 2019). Provable (provable, 2019). Witnet (Adan & al., 2017). Astrea (Adler et al., 2018). Augur (Peterson & al., 2015). Eternity (Hess & al., 2017). Chainlink (Ellis et al., 2017). PriceGeth (Eskandari & al., 2017). Majority is not enough (Ittay & Emin., 2018). TrustedAP (Fox., 2021). Unsecured smart contract detection (Kevin & al., 2021).
	Tracing vulnerabilities from a large number of contracts	Long Short-Term Memory - LSTM (Wesley et al., 2019). Bidirectional LSTM (BLSTM) (Peng & al., 2020). BLSTM + Attention (BLSTM-ATT) (Peng & al., 2020). AWD-LSTM (Ajay et al., 2020). S-gram scheme (Zibin & al., 2019). A Novel Machine Learning-Based Analysis Model (Xu & al., 2021).
	Integer overflow/ Underflow	Easyflow (Gao & al., 2019). Oyente (Loi & al., 2016). Zeus (Sukrit et al., 2018). ReGuard (Chao et al., 2019). S-gram (Zibin et al., 2019; Han et al., 2018). ContractGuard (Wang & al., 2020). EVMPatch (Rodler & al., 2020). Osiris (Christof & al., 2018). SolAnalyser (Akca & al., 2019). SIF (Chao et al., 2019). Formal Verification Framework (Tianyu & Wensheng, 2020). Elysium (Torres & al., 2021). Eth2vec (Ashizawa & al., 2021). HFContractFuzzer (Ding & al., 2021).
	Suicidal, prodigal and greedy contracts	Maian (Ivica et al., 2018). LSTM (Wesley et al., 2019). Fuzzer symbolic execution (Jingxuan & al., 2019). Teether (Johannes and Christian., 2018). EVMPatch (Rodler et al., 2020). TxSpector (Mengya & al., 2020). Ethbmc (Frank et al., 2020). Elysium (Torres & al., 2021). Alternate authentication with smart contract (Boron & Kobusińska., 2021).

continued on following page

Table 2. Continued

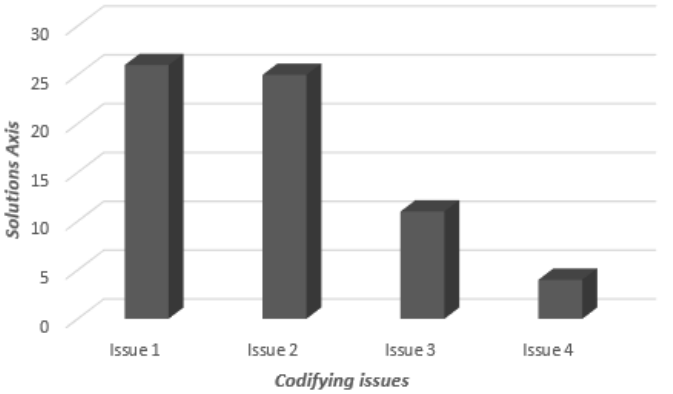
Smart Contract Problems		Techniques/Developed Tools
Privacy issues	Lack of transactional privacy	Encryption techniques (Zether, Secret store, and More secrets) (Hiroki et al., 2015; Benedikt et al., 2019; Rachel, 2018). Time-locked secrets and Secret contracts (Rachel., 2018). ShadowEth (Yuan & al., 2018). Privacy Preserving Solutions (Jorge & al., 2019). Privacy Guard (Yang & al., 2020). Access control privacy framework for DOSNs (Rahman et al., 2019). SMACS (Liu & al., 2020). Use of artificial intelligence (Rajesh & al., 2020). Private data object, Ekiden (Cheng & al., 2019). Fastkitten, Eli, Teechain, Quorum, Zocrates, and Zexe (Hu & al., 2021). Privacy-Preserving Healthcare Platform (Omar & al., 2021). Deep Blockchain Framework (Al-Kadi & al., 2021). Smart Contract-Based Blockchain-Envisioned Authentication Scheme (Vangala & al., 2021). smartFHE (Solomon & Almashaqbeh., 2021). TrustedAP (Fox., 2021). Blockchain-based smart contract framework (Vardhini & al., 2021). A Blockchain-based Framework for Information Management in Internet of Vehicles (Wen & al., 2021). Blockchain-independent smart contract infrastructure (Saquib & al., 2021). blockauth (Zhaofeng & al., 2021). Fortified-Chain (Egala & al., 2021). Reputation management smart contract (RM) (Geng & al., 2021). Fasten (Damle & al., 2021). Privacy Preservation for On-Chain Data (Ziar & al., 2021). SmartMedChain (El Majdoubi & al., 2021). Intelligent Mediator-based Enhanced Smart Contract (Mucheol & Junho., 2021). TREAD (Khan & al., 2021). Privacy preserving platform for COVID-19 vaccines (Barati & al., 2021). TeSC (Gallersdörfer & Matthes., 2021).
	Data feed privacy issues	Town Crier (TC) (Fan & al., 2016). Practical Data Feed Service (PDFS) (Juan & Pawel., 2018). TrustedAP (Fox., 2021). A Blockchain-based Framework for Information Management in Internet of Vehicles (Wen & al., 2021).
Performance issues	Smarts contract sequential execution problems/scalability issues/single point of failure/efficiency	Parallel execution of smart contracts (techniques adapted from software transactional memory and optimistic Software Transactional Memory systems (STMs)) (Vukolić, 2017). Concurrent execution of different contract models (Massimo et al., 2020; Wei et al., 2018). Arbitrum and Yoda . Asynchronous and Concurrent Execution (ACE) (Karl et al., 2019). OV: Validity-based optimistic Smart contract (Quan et al., 2020). Two-phase concurrency control protocol . Blockumulus . Cloak (Ren et al., 2021). Escort . Hybrid smart contract architecture . Smart contracts for automated control system . Blockchain-based smart contract framework . A Blockchain-based Framework for Information Management in Internet of Vehicles . SCBAC . Blockchain-independent smart contract infrastructure . Automating Procurement Contracts . Blockauth . Blockeye . Clock finance . Graph Neural Network with expert knowledge . Map-reduce based parallel computation . EtherSolve . Fortified-Chain . Two-phase framework based on trusted hardware Intel SGX . Deserving resource smart contract (DRSC) . Smart Contracts for Verifying DNN Model Generation Process . Flexible Smart Contract Interaction Framework with Access Control (FSCC) . *FASTEN* . *EtherClue* . TREAD . Smart contract sharding .
	Contract redeployment efficiency	Decompilation capabilities encapsulated (Santiago et al., 2018).

Figure 2. Relationship Between Issues and Solutions



In codifying issues, under-optimized contracts and contract writing difficulties are the most tackled issues. Indeed, From figure 3, more than 20 solutions exist for each. For smart contract programming language and smart contract termination problems, we denote 11 and 4 solutions, respectively. This shows how important some issues are concerning others.

Figure 3. Codifying Issues Diagram



Unoptimized smart contracts waste money through gas consumption. Thus, as money is involved, care must be taken to provide a flexible and extensible environment that can easily implement efficient contracts, which should be less gas-costly. Concerning unoptimized smart contract solutions to tackle the issue, SmartCheck (Sergei & al., 2018), MadMax (Neville & al., 2018), Maian (Ivica N. & al., 2018), Securif (Petar & al., 2018), SmartCopy (Yu & al., 2019), Gas reducer (Ting & al., 2018), Gastap (Sara & Ralph., 2019), Gasper (Bill and Ari., 2016), Gasol (Elvira & al., 2019), GasChecker (Ting et al., 2020), GasFuzzer (Ashraf et al., 2020), SafeVM (Elvira & al., 2019), SolAnalyser (Akca & al., 2019), Mythril (Sarwaar et al., 2020), contract's versions comparisons, Graph Neural Networks (Yuan & al., 2020), sCompile (Jialiang et al., 2019), SCRepair (Yu et al., 2020), Formal Verification framework (FVF) (Maher & Aad Van, 2017; Jing & Zhentian, 2019), SmartShield (Yuyao et al., 2020), Manticore (Mark & al., 2019), Zeus (Sukrit et al., 2018), Echidna (Gustavo & al., 2020) can be used. Eth2vec (Ashizawa & al., 2021) and Machine Learning Approach for Gas Price Prediction in Ethereum Blockchain (Mars & al., 2021) are other methods to detect such contracts.

Regarding correct smart contract programming language and complexity of programming language, the easier the process of writing contracts, the less the contract will be subjected to errors. The correctness of smart contracts ensures the right functioning of the contracts, meaning the contracts should carry out actions as intended by programmers. While the faithful execution of smart contracts depends on Blockchain's consensus protocol, participating entities should ensure the contract's fairness and its correctness remain a major concern as part of their prerogatives (Sukrit & al., 2018). The importance of this comes from the fact that money is involved, and any mistake could result in disastrous consequences. A particular example is the Distributed Autonomous Organization (DAO) attack, where a massive loss of money occurred (Maher A. & Aad Van M., 2017). Several solutions are proposed to address this issue including Adoption of Formal verification methods (Maher. & Aad Van., 2017; Jing & Zhentian., 2019; Tesnim & Kei-Leo., 2018), Annotary (Konrad and Julian., 2019), Guidelines/standard for developers (Kevin et al., 2016; Bartoletti & Pompianu, 2018), ContractWar (W. Wang et al., 2020), ContractGuard (X. Wang X. et al., 2020), SmartDEAMP (Zibin & al., 2019), Gigahorse (Zibin & al., 2019), Semi-automation of smart contract creation (Christopher & Mariusz ., 2016), New contract language development (Jing & Zhentian., 2019), Semantics analysis (Zibin et al., 2019; Anastasia and Aaron., 2018), Raziel (David, 2020), Solidifier (Antonino & Roscoe., 2020), Verismart (Sunbeom & al., 2019), VeriSolid (Anastasia & Aron., 2019), Osiris (Christof & al., 2018), SASC (Zhou et al., 2018), SCRepair (Xiao et al., 2020), SmartShield (Yuyao et al., 2020), Manticore (Mark & al., 2019), FSolidM (Anastasia and Aaron., 2018), VerX (Anton et al., 2020), and Zeus (Sukrit et al., 2018).

Therefore, the choice of the programming language plays a significant role as the choice must be made according to the contract objective, flexibility, and to prevent unnecessary tasks that would be done with a difficult language choice. Initially, procedural languages are used to write current smart contracts, and the Solidity language is an example. However, procedural language requires that the execution code follows a series of steps specified by developers. As such, what should be done and how to do it must be clearly defined to avoid any misbehavior of the contract. As a result, writing smart contracts using this type of language becomes complex and subject to errors (Florian & al., 2016).

Several approaches are then proposed to ease smart contract development: Logic-based languages (Prolog) (Zibin et al., 2019; Florian et al., 2016), IELE (Scillia, Yul) (Tyurin & al., 2019), Use of type based-language Idris, Simplicity (O'Connor R., 2017), liquidity (Çagdas & al., 2018), Obsidian (Coblenz, 2017), Flint (Schrans & al., 2018), Mandala (Markus, 2019) SmaCoNat (Regnath & Steinhors ., 2018), Bitml (Tyurin & al., 2019), SPESC (Xiao & al., 2018), iContract (Qasse & al., 20201), Smart-Graph (Pierro., 2021), and SuMo (Barboni & al., 2021).

Smart contract termination or modification is the least tackled issue. However, solutions have been given. According to the authors, a set of norms can allow smart contract modification and termination. It involves taking legal contracts' rules or standards and redefining them to go along the smart contract.

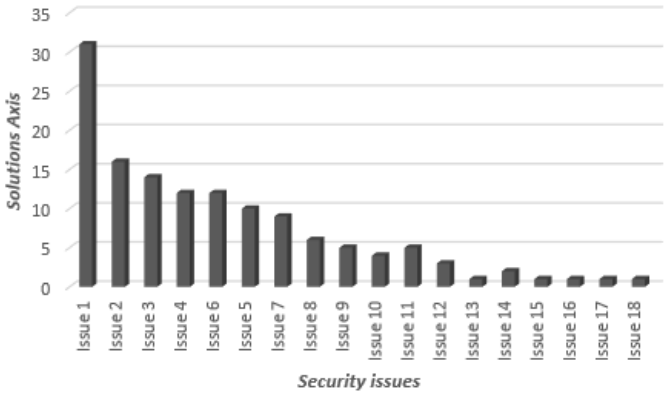
Mechanized termination proof (Thomas et al., 2020) is another way to address the issue and is based on the EVM abstract model. An internal counter is used to evaluate the contract's termination independently of gas system presence. This type of contract termination is used in EVM contracts and written in EVM bytecode. Other solutions such as proof-carrying smart contracts (Hu & al., 2021), and Intelligible Description Language Contract (IDLC) are also provided.

SECURITY ISSUES

Figure 4 shows the number of provided solutions against security issues.

Issue 1 = Reentrancy vulnerability
Issue 2 = Mishandled exception vulnerability
Issue 3 = Integer overflow/underflow
Issue 4 = Transaction-ordering vulnerability
Issue 5 = Timestamp vulnerability
Issue 6 = Untrustworthy data feed issues

Figure 4. Security Issues Diagram



Issue 7 = Social, prodigal, and greedy contracts
Issue 8 = Tracing vulnerabilities from a large number of contracts
Issue 9 = Delegated puzzles
Issue 10 = Ponzi scheme issues
Issue 11 = Criminal/opaque smart contract activities
Issue 12 = Block randomness
Issue 13 = BGP routing concerns
Issue 14 = HoneyPot
Issue 15 = Selfish Mining
Issue 16 = Verifier's dilemma
Issue 17 = Denial of service with block stuffing

Security issues are the ones that cause severe damages to the blockchain ecosystem with the DAO and multi-Sig parity wallet. They are vulnerabilities or threats someone might exploit to attack the blockchain system or use it to get funds from victims (Maher A. & Aad Van M., 2017).

According to Figure 4, reentrancy is the the most tackled problem is one of the notorious issues that caused a huge loss of funds. The issue occurs when several repetitive withdrawals are performed by a malicious user who uses a recursive call function while his balance is only deduced once (Maher A. & Aad Van M., 2017). It ranks with more than 20 solutions provided for its resolution while improvements are still needed.

These solutions ÆGIS (Torres & al., 2020), ReGuard (Chao & al., 2018), Reentrancy analyzer (Chinen & al., 2020), Bidirectional LSTM – ATTention (BLSTM-ATT) (Peng & al., 2020), ECFChecker (Grossman & al., 2017), EthScope (Wu & al., 2020), Sereum (Michael & al., 2019), Teether (Johannes and Christian., 2018), Zeus (Sukrit et al., 2018), Graph Neural Networks (Yuan & al., 2020), SmartCheck (Sergei et al., 2018), Securify (Petar & al., 2018), SmartCopy (Yu & al., 2019), Manticore (Mark & al., 2019), Mythril (Sarwaar et al., 2020), Oyente (Loi & al., 2016), ContractFuzzer (Jiang & al., 2018), ContractWar (Wang et al., 2020), TxSpector (Mengya & al., 2020), Vandal, (Brent & al., 2018) Slither (Josselin & al., 2019) are able to detect reentrancy bugs as well as Mechanism to Detect and Prevent Ethereum Blockchain Smart Contract (Alkhalifah & al., 2021), Dynamit (Eshghie & al., 2021), Elysium (Torres & al., 2021), Eth2vec (Ashizawa & al., 2021), Reentrancy detection using TXL programming language (Samreen & Alalfi., 2020), Graph Neural Network with expert knowledge (Liu & al., 2021), EtherSolve (Contro & al., 2021), SGuard (Nguyen & al., 2021).

Mishandled exception, integer overflow/underflow, and transaction ordering followed, respectively, with more than 10 solutions provided for each. A mishandled exception is the lack of communication between two contracts that causes the problem (Maher A. & Aad Van M., 2017). A typical example is the occurrence of a lack of gas (currency units necessary for contract execution and rewarding miners) exception in the callee's contract, which is not propagated to the caller based on the call function creation. Then, there is a possibility of causing trouble due to the non-reported exception. As solutions, we have Check of the return value (Loi & al., 2016), ContractFuzzer (Jiang & al., 2018), Vandal (Brent & al., 2018), Zeus (Sukrit et al., 2018), contract versions comparison, SmartCheck (Sergei & al., 2018), Security (Petar & al., 2018), SmartCopy (Yu & al., 2019), Mythril (Sarwaar et al., 2020), ContractWar (Wang Wei & al., 2020), Slither (Josselin & al., 2019), Oyente (Loi & al., 2016), Fuzzer symbolic execution (Jingxuan & al., 2019), EVMPatch (Rodler & al., 2020), SolAnalyser (Akca & al., 2019), Elysium (Torres & al., 2021).

Integer overflow/underflow occurs in arithmetic operations. An over or under issue occurs when a procedure is performed and will need a fixed variable size to keep an operand (e.g., uint256), which is beyond the data type of the variable (e.g., uint8). The calculation then oversteps the upper band value or is down the lower band value as max + 1 ® min or min − 1 ® max (Wang Wei & al., 2020). Several tools can solve this issue: Easyflow (Gao & al., 2019), Oyente (Loi & al., 2016),

Zeus (Sukrit et al., 2018), ReGuard (Chao et al., 2019)., S-gram (Zibin et al., 2019; Han et al., 2018), ContractGuard (Wang & al., 2020), EVMPatch (Rodler & al., 2020), Osiris (Christof & al., 2018), SolAnalyser (Akca & al., 2019), SIF (Chao et al., 2019), and Formal Verification Framework (Tianyu & Wensheng., 2020) can find this vulnerability as well as Elysium (Torres & al., 2021) and Eth2vec (Ashizawa & al., 2021).

Regarding the transaction ordering issue, the occurrence of this problem is related to transactions that call upon the same contract while a block contains all of them. A disarranged execution order of transactions might result in an attack van if it is not planned carefully (Loi & al., 2016). As for solutions, Ethereum Based Functions (use of SendIfReceived function) (Christopher & Vincent., 2016), Oyente (Loi & al., 2016), Ether racer (Aashish & al., 2019), Fuzzer symbolic execution (Jingxuan & al., 2019), FSolidM (Anastasia and Aaron., 2018), ContractWar (Wang Wei & al., 2020), Zeus (Sukrit et al., 2018), Securify (Petar & al., 2018), Mythril (Sarwaar et al., 2020).

Time-stamped, untrustworthy data feeds and suicidal, prodigal, greedy contracts also received much attention, with several solutions developed.

Time-stamped vulnerability is a time dependency contract is the one that utilizes block timestamp as a triggered condition to transactions executions (e.g., sending money) (Maher A. & Aad Van M., 2017). The current local time is the reference time used by the miner to set the block timestamp of the block it generated. But it can choose a timestamp condition value that can favor him because there is a possibility of causing inaccuracy in a block timestamp. Malicious miners can vary the timestamp value from the current time and ensure block acceptance in the blockchain system. This results in the possibility of favoring users in the case of activities such as games (e.g., TheRun Contract) where the timestamp is an important parameter in deciding the winner of the jackpot. As solutions to solve this issue, we have Random seed of block number (Loi & al., 2016), SIF (Chao & al., 2019), Slither (Josselin & al., 2019), SmartCopy (Yu & al., 2019), ContractWar (Wang et al., 2020), Mythril (Sarwaar et al., 2020), Graph Neural Networks (Yuan & al., 2020), Fuzzer symbolic execution (Jingxuan & al., 2019), SolAnalyser (Akca & al., 2019), Eth2vec (Ashizawa & al., 2021), SmartPulse (Stephens & al., 2021), and Graph Neural Network with expert knowledge (Liu & al., 2021).

Untrustworthy data feeds are derived from the external feeding of the Blockchain for some smart contracts and guarantee is not assured for the imported data. Solutions exist to tackle the issues: town Crier (TC) (Fan & al., 2016), Reputation contract (Zibin & al., 2019), Provable (provable, 2019), Witnet (Adan & al., 2017), Astrea (Adler et al., 2018), Augur (Peterson & al., 2015), Eternity (Hess & al., 2017), Chainlink (Ellis et al., 2017), PriceGeth (Eskandari & al., 2017), Majority is not enough (Ittay & Emin., 2018), and TrustedAP (Fox., 2021).

Contracts that are killed from anonymous addresses are known as suicidal contracts. Most of the time, a contract has a security option that enables it to be killed by its owner (or trusted addresses) in case of attacks or malfunctions. A malicious user can also exploit this option to cause trouble (Wesley et al., 2019; Ivica, 2018). Prodigal contracts permit fund leakage to unknown accounts. Contracts often have internal calls to return money to their owner when attacked (Ivica, 2018), which is exploited to perform this vulnerability. Greedy contracts are alive contracts that can't release ether. A library responsible for ether withdrawing is killed. Many accounts dependent on the killed library involved in the ParitySig attack contract were unable to release funds. The following tools can tackle the issues: Maian (Ivica et al., 2018), LSTM (Wesley et al., 2019), Fuzzer symbolic execution (Jingxuan & al., 2019), Teether (Johannes and Christian., 2018), EVMPatch (Rodler et al., 2020), TxSpector (Mengya & al., 2020), Ethbmc (Frank et al., 2020), and Elysium (Torres & al., 2021).

For vulnerability tracing issues, delegated puzzles, Ponzi schemes, and criminal/opaque contract activities, solutions are provided for their respective resolutions.

In vulnerability tracing issues, invocation depth is the main problem. Indeed, as specific symbolic tools use complex analysis steps, a predefined invocation depth is also needed to look for exposed execution paths. As the depth goes up, so does the search time (Wesley J. T & al., 2019). Therefore, the searching time will be so high that specific contracts cannot be analyzed from a certain number

of smart contracts. The failure of these tools to analyze the increasingly large number of contracts that we have in the Ethereum platform might lead to faulty contracts that can cause losses. These solutions are useful to protect smart contracts: Long Short-Term Memory - LSTM (Wesley et al., 2019), Bidirectional LSTM (BLSTM) (Peng & al., 2020), BLSTM + Attention (BLSTM-ATT) (Peng & al., 2020), AWD-LSTM (Ajay et al., 2020), S-gram scheme (Zibin & al., 2019), a Novel Machine Learning-Based Analysis Model (Xu & al., 2021).

Delegated puzzles (Wenbo et al., 2019) are problems where a cunning miner will break a searching puzzle solution work into several smaller works and later delegate them to outsourcers which might be untrusted workers. This will lead to the possibility of accumulating several puzzle works and delegating them, and increasing its revenue when its workers solve all puzzles. As for the solution, piece of work (Philip & al., 2017), Non-delegated Scratch-Off Puzzles (Miller & al., 2015), Sign to Mine (Ziftr., 2014), Phase-Proof of Work (2P-PoW) (Eyal & Sirer., 2014) can be used.

Ponzi schemes are fraudulent investment systems that are established and provide a high return with little or no risk (Massimo & al., 2018). Funds paid by new investors are used to pay old investors, and the system stops when no new investors come in. Several machine learning techniques are dedicated to the issue. Datasets are constructed using real blockchain transactions and are used to train classification models for threats detection. Several Machine learning techniques (Ripper, Bayes Network and Random Forest, and XGBoost) are used for their detection (Weili & al., 2018).

Criminal/opaque contract activities can be conducted through smart contracts, making them criminal smart contracts (CSC) (Ari & al., 2016). The authors identified three different types of criminal activities, namely, leakage/sale of secret documents, theft of private keys, and calling-card crimes (murder, arson, etc.). Leakage of secret documents is related to the public disclosure of secrets, leading to payments if the data is provided within the right time. A key-theft contract might be commission-fair if its perpetrator gets rewarded for delivering the private key that he stole, which must be valid within a certain period. A calling card is an unpredictable feature of a premeditated crime. When combined with authenticated data feeds, it can support many CSC (Criminal Smart Contracts). Crimes are executed, and codes are used to ensure the veracity of the crime before rewarding. The resolution of these issues is important to promote a secured and crime-free smart contract ecosystem. Trustee Neutralizable smart contract (Ari & al., 2016), Erays (Yi & al., 2018), Smart Inspect (Santiago et al., 2018), Samos (Knecht & Stiller., 2021), and smart contract-based Supply Chain Control (Dietrich & al., 2020) are used to tackle the issue.

Regarding denial of service, the DoS block stuffing (Crypto P., 2018) is a blockchain-based Smart contract threat that permits a malicious user to give high Gas Price incentives to miners for his transactions to be taken care of in new blocks to the detriment of other blocks. Smart Contract-Based Solution for Secure Distributed SDN (Almakhour & al., 2021), SmartScan (Samreen & Alalfi., 2021), and Co-Chain SC (Houda & al., 2019) have tried to solve the problem.

Considering block randomness, BGP routing, honey pot, selfish mining, verifier's dilemma, and the man in the middle, only one solution is developed for each.

Considering block randomness, there is the possibility that block generation and release are based on the miner's will while profit is at stake. Therefore, randomness is compromised, and the system becomes tricky (Zibin & al., 2019). To solve it, we have the Delay and Sloth function (Zibin et al., 2019) introduced in the contract to avoid the execution of the miner's will.

In this issue, Border Gateway Protocol (BGP) routing scheme is the main asset in capturing blockchain information (Maria A. & al., 2017). This can result in a high broadcasting delay of data or messages, traffic hijack, and digital currency robbery. The SABRE is a proposed solution that adjusts BGP routing policies from several domains while protecting the link between clients and relays through good relay alignment, appropriately placing relays. The same sabre network uses hardware and software co-designing in software-defined networking (SDN) to cut down relays traffic.

Honey pots are traps hidden within smart contracts where certain conditions of the contract will require funds from users while not providing an expected result. Attackers use several techniques

further developed in (Christof & Mathis., 2019) work. The issue is tackled with HoneyBadger, which uses symbolic execution and heuristic techniques to detect this vulnerability. The tool takes in a smart contract in bytecode and outputs a final result where all honey pot techniques found are stated.

Selfish Mining (Cyril & Ricardo, 2019) is a security threat where an attacker generates several private blocks and broadcasts them one by one in the Blockchain or keeps them hidden and reveals them to the public at the right time to increase its incentives or rewards (uncle blocks rewarding). A proposed solution in the Bitcoin network (Ittay & Emin., 2018) is modifying the blockchain protocol that would allow pools to command less than one quarter of the available resources. However, in the Ethereum network, solutions are still in the infancy stages as the problem is more complicated (Cyril & Ricardo., 2019).

Verifier's Dilemma (Loi et al., 2015) is a threat that forces miners to accept unvalidated blockchains in exchange for high gas incentives or to waste resources as the computational effort required to validate some blocks is very demanding, blocking them in the race o mining next blocks. It creates an atmosphere where greedy miners intentionally block honest miners to get higher rewards. A proposed solution is creating a consensus computation framework that achieves correctness by sharing computation tasks across several blocks of transactions that comply with the ε-consensus computer model. Thus, verification cost is reduced across multiple blockchain transactions.

Man in the middle is security issue based on authentication problems in organizations where people are getting access to resources cannot be securely verified. Users are given roles to only access resources, which can become challenging to address before user authentication and role issuance. To address these issues, a Role-Based Access Control (RBAC) system is developed to resist a man in the middle where attackers cannot forge

digital signatures of others without their private keys, which helps in preventing the problem.

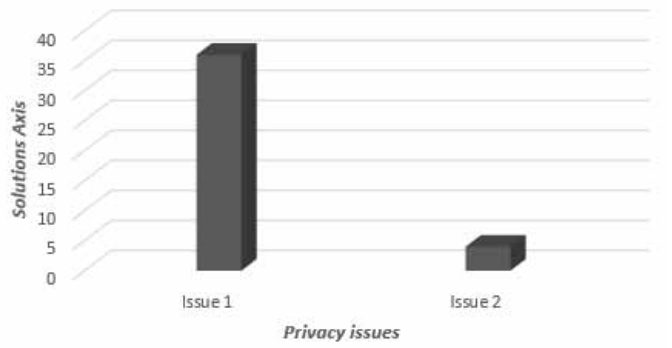
PRIVACY ISSUES

Figure 5 represents privacy concerns against their developed solutions.

Issue 1 = Lack of transactional property
Issue 2 = Data feeds privacy issues

Privacy concerns are problems that are derived from contract information exposure to the public (Yang & al., 2020). Two problems are identified: the lack of transactional privacy and data feed privacy (Maher A. & Aad Van M., 2017). Transactional privacy is more frequently tackled than data feed privacy issues for privacy issues. According to Figure 5, the former produced more than 25

Figure 5. Privacy Issues Diagram



solutions but the latter, only four. Data should be taken care of as attacks can be performed through imported data (Juan & Pawel., 2018).

Regarding the lack of transactional property, encryption techniques applied to

smart contracts before deployment are the most obvious ways to protect smart contracts. They are proposed by (Hiroki W. & al., 2015) and the contract access is restricted to people involved in the contract who have their decryption keys. Other ways to allow contract privacy are Zether, Secret store, and More secrets. They are all based on encryption/decryption techniques (Benedikt B. & al., 2019) (Rachel., 2018). Other solutions include private data object, Ekiden (Cheng & al., 2019), Fastkitten, Eli, Teechain, Quorum, Zocrates and Zexe (Hu & al., 2021), Privacy-Preserving Healthcare Platform (Omar & al., 2021), Deep Blockchain Framework (Al-Kadi & al., 2021), Smart Contract-Based Blockchain-Envisioned Authentication Scheme (Vangala & al., 2021), smartFHE (Solomon & Almashaqbeh., 2021), TrustedAP (Fox., 2021), Blockchain-based smart contract framework (Vardhini & al., 2021), A Blockchain-based Framework for Information Management in Internet of Vehicles (Wen & al., 2021), blockchain-independent smart contract infrastructure (Saquib & al., 2021), blockauth (Zhaofeng & al., 2021), Fortified-Chain (Egala & al., 2021), Reputation management smart contract (RM) (Geng & al., 2021), and Fasten (Damle & al., 2021).

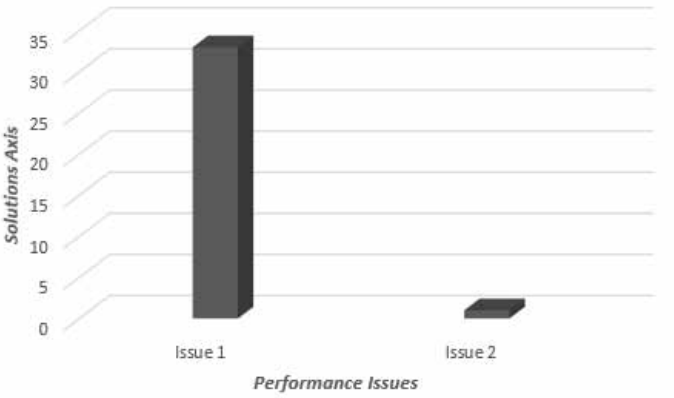
For data feed privacy issues, the problem arises because all the data feeds needed by a contract to operate are exposed to the public (Fan & al., 2016). One of the solutions, Practical Data Feed Service (PDFS), is a system that securely connects content providers with their Blockchain (Juan & Pawel., 2018). The authentication of data is provided over Blockchain without affecting trust chains. Providers' contents are easily parsed and converted into a different usable format. The system provides security, transparency, efficiency, and auditability of content providers while reducing their spiteful deeds. Town Crier (TC) is another solution

built by (Fan & al., 2016) that performs as a bridge between smart contracts and outside data providers. It helps in providing authenticated data feeds for smart contracts. A contract uses the TC's public key to encrypt a request, and upon reception, the encrypted request is decrypted with the TC's private key. This process secures the requested content from people/contracts not involved in the data feed. Other solutions are TrustedAP (Fox., 2021) and a Blockchain-based framework for information management on the internet of vehicles (Wen et al., 2021).

PERFORMANCE ISSUES

Figure 6 represents the relationship between the issues and the solutions.

Figure 6. Performance Issues Diagram



The main issue in blockchain technology is the sequential execution of smart contracts (Vukolić M., 2017). Smart contracts in blockchain systems are executed one after the other. As the number of contracts is drastically increasing, running all those contracts sequentially will affect the whole performance, as noticed by (Maher A. & Aad Van M., 2017) because smart contract execution within a period will be very limited. Performance is always on the moving track in systems to satisfy the system user. As such, improvement is always necessary to cope with reality. According to Figure 6, more than 30 solutions have been developed to tackle its problem.

Sequential execution, scalability, single point of failure, and efficiency have been tackled differently in smart contract execution. Solutions are based on concurrent smart contract execution techniques that depend on multi-thread technology with transaction-splitting algorithms and the use of an optimistic Software Transactional Memory system (STMs). Other solutions developed include Arbitrum and Yoda (Hu et al., 2021), two-phase concurrency control protocol (Jin & al., 2021), Blockumulus (Ivanov & al., 2021), Cloak (Ren et al., 2021), Escort (Lutz & al., 2021), hybrid smart contract architecture (Solaiman & al., 2021), Smart contracts for automated control system (Pradhan & Singh., 2021), Blockchain-based smart contract framework (Vardhini & al., 2021), A Blockchain-based Framework for Information Management on Internet of Vehicles (Wen & al., 2021), SCBAC (Song & al., 2021), blockchain-independent smart contract infrastructure (Saquib & al., 2021), Automating Procurement Contracts (Omar & al., 2021), blockauth (Zhaofeng & al., 2021), Blockeye (Wang & al., 2021), Clock finance (Babel & al., 2021), Graph Neural Network with expert knowledge (Liu & al., 2021), Map-reduce based parallel computation (Muchhala & al., 2021), EtherSolve (Contro & al., 2021), Fortified-Chain (Egala & al., 2021), Two-phase framework based on trusted hardware Intel SGX (Fang & al., 2021), Deserving resource smart contract (DRSC) (Yang & al., 2021), Smart Contracts for Verifying DNN Model Generation Process (Seike & al., 2021), Flexible Smart Contract Interaction Framework with Access Control (FSCC) (Li & Asaeda., 2021), and Fasten (Damle & al., 2021).

As smart contract usage increases exponentially with contracts being redeployed from time to time, further improvement is required. Binary decompilation reduces redeployment cost but is still in its infancy stage, as only one solution has been produced. The decompilation technique allows a particular method that introspects the smart contract's current state without the redeployment of the smart contract (Santiago et al., 2018).

In summary, irrespective of the category of issues, lack of transactional privacy and other problems such as sequential execution and scalability, reentrancy, correct smart contract writing, and under-optimized contracts are the most recurrent problems that are tackled. Indeed, these problems lead to data exposure to the public and loss of money, which are valuable assets to be preserved seriously. Comparing reentrancy and parity multi-sig (suicidal contract), they both caused serious financial consequences. However, we can notice that reentrancy is far beyond tackled as compared to parity multi-sig wallet and this might be related to its severity. Problems with one solution, such as Delagated puzzle, HoneyPot, Bgp routing, blockrandomness, verifier dilemma, etc., are less common in the literature, perhaps because they are less severe and would need to be reconsidered for a different approach of solutions.

DISCUSSION

Implications for Practice

This study references smart contract issues related to coding, security, privacy, and performance. Solutions are categorized according to each problem. The first implication is that it helps Blockchain-enabled smart contract users to get tools concerning a particular issue. Developers can easily know the appropriate tool to be used to check a particular issue in the work they are developing.

Secondly, from a smart contracts point of view, the right coding is paramount as money is involved in most systems. This study provides different ways, such as the use of appropriate guidelines and Logic-based languages (Prolog) (Zibin et al., 2019; Florian et al., 2016), IELE (Scillia, Yul) (Tyurin & al., 2019), and the use of type based-language, to name a few, to ensure an error-free smart contract with more straightforward writing language. They all help develop smart contracts and ease smart contract development. Furthermore, they provide more minor gas usage systems as smart contracts can be quickly terminated and optimized with the available tool. Thirdly, looking at the divergence of tools for a particular issue gives practitioners the possibility of changing means if difficulties are found with a particular tool. As a result, smart contract users might be more confident in using the technology, though improvement is still needed as some issues are yet to be solved.

IMPLICATIONS FOR RESEARCH

This research provides an interesting reflection for researchers. Furthermore, it gives a new framework in researching hot topics regarding smart contract safety will be discovered. For example, privacy and performance issues are left behind compared to security and codifying issues, and a study might be conducted to find out new considerations concerning them.

Most of the tools have been developed using different techniques or approaches, and a new area of research could be to dig further into their respective performance. Furthermore, as this study has explored security, codifying, privacy, and performance issues, other studies can be conducted to explore other aspects of smart contracts, such as life cycle or function based on blockchain architecture layers.

LIMITATIONS

This study has several limitations concerning issue classifications and solutions. First, the study does not dig deeply into the cause of each issue. This may reduce the comprehension related to each category of issue. Secondly, the study does not find if these problems are common to all blockchain-enabled smart contracts such as Ethereum and Tron, or if some blockchains are more affected than others, as the technology used varies from one Blockchain to another creation. Third, each category of issues could be further extended. Indeed, we can match each category of issues to a particular layer of blockchain-enabled smart contracts, thus emphasizing the most vulnerable layer. Fourth, there is a need for more research to be conducted to increase the existing problems-and-solutions portfolio. The study could go beyond coding, security, privacy, and performance issues and solutions, and offer problems related to dynamic and static execution of tools with respective approaches. Indeed, as technology evolves, new threats might be discovered. However, establishing such research with a realistic scenario could be very time-consuming and not covered in this study. Fifth, some solutions can tackle several issues, but their accuracy and efficiency are still a concern as no measurement tools are used in this study to evaluate those parameters. Thus, each tool's most relevant performance area is not provided in this review. Finally, a road map for the most accessible tool used in each category when tackling a particular concern could be emphasized in this study. However, it's not covered as tool classification (distinguishing between private and public tools) is not tackled in this study to help save time when looking for a particular one for a typical issue.

OPEN RESEARCH CHALLENGES

Our study has led to the definition of new challenges that can enhance smart contract security. We consider the different domains in this study and solutions to solve each problem.

CODIFYING ISSUES

Developers face many difficulties in writing smart contracts. Smart contracts embrace all domains, and several languages have been created to ease contract creation. A thorough study of these languages will help emphasize the easiest and least prone to artificial error to be used in a particular domain, such as the health care system (Tyurin & al., 2019). Further investigation can consider Turing's complete contract language limitation, mainly used to develop smart contracts (Marc & al., 2019) to improve its effectiveness. The complexity of a solidity-based smart contract can be measured with static metrics. But this is still in its early stages and will require further investigation to set a new benchmark for smart contract programming. Some Gas estimation tools (Gastap) prevent gas error dynamically with the support of Oyente tools. However, further research can search for the best smart contract gas estimation tool (Sara & Ralph., 2019). Finally, as the smart contract is not bound to modification after deployment, it will be relevant to deploy a test environment for testing its accuracy before real-world usage.

SECURITY ISSUES

Several studies have been conducted to solve the issues. However, few research types using deep-learning techniques to tackle those issues have been conducted, raising the need to explore that possibility further. As data are imported into the Blockchain, irrelevant, malicious, or erroneous data can be among the imported data. New strategies or tools should be developed to preserve attacks by thoroughly analyzing the imported data (Hamda & al., 2020). Other issues such as delegated puzzles, denial of services, selfish mining, and Verifier's dilemma are still open research areas (Huashan & al., 2020) and need more attention for smart contract protection.

PRIVACY ISSUES

Artificial intelligence is a new era that is yet to be explored. Further directions might focus on protecting data, especially health data, as they are very sensitive. Other researchers may embrace analysis and decision making to classify data or protect users (Rajesh & al., 2020).

PERFORMANCE ISSUES

Scalability is still an issue that needs to be ameliorated as smart contract usage increases. Concurrent executions of smart contracts can be improved with new languages that rely on static analysis of reading/written key mechanisms to increase concurrency degree (Massimo & al., 2020). Other studies may tackle the complexity between child chains and the main child regarding Ethereum scalability.

CONCLUSION

This study provides a global view of blockchain-based smart contract issues, mainly regarding smart contract performance, privacy concerns, codifying, and security. We downloaded several papers and also identified new threats that fall into one of the categories. The corresponding solutions to the new threats are provided, and they can also be used for existing threats. We noticed that some threats have not been addressed or have been addressed with only a few solutions, and need more attention from the scientific community. Finally, we finally provide research directions so that smart contracts get secured in the future.

FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.

ACKNOWLEDGMENT

This work has been supported by the African Center of Excellence in Mathematics, Informatics and Applications (ACE-MIA) project in Benin. The authors would also like to thank Prof. Samuel Fosso Wamba (Toulouse Business School, France) and Dr. Thierry Edoh for their invaluable suggestions.

REFERENCES

Aashish, K., Ivica, N., Ilya, S., Aquinas, H., & Prateek, S. (2019). Exploiting the laws of order in smart contracts. *Proceedings of the 28th ACM SIGSOFT International Symposium of Software Testing and Analysis*, 363-373.

Adler, Berryhill, Veneris, Poulos, Veira, & Kastania. (2018). Astraea: A decentralized blockchain oracle. *Proceedings of IEEE International Conference Internet Things (iThings) IEEE Green Computing and Communications (GreenCom) IEEE Cyber, Physical Social Computing and Networking (CPSCom) IEEE Smart Data (SmartData)*, 1145–1152. doi:10.1109/Cybermatics_2018.2018.00207

Ajay, Swayamjyoti, Sahoo, Sahu, & Kishore. (2020). *Multi-Class classification of vulnerabilities in Smart Contracts using AWDLSTM, with pre-trained encoder inspired from natural language processing.* arXiv:2004.00362.

Akca, S., Rajan, A., & Peng, C. (2019). SolAnalyser: A Framework for Analysing and Testing Smart Contracts. *Proceedings in the 26th Asia-Pacific Software Engineering Conference (APSEC)*, 482-489. doi:10.1109/APSEC48747.2019.00071

Al-Kadi, O., Moustafa, N., Turnbull, B., & Choo, K. (2021). A Deep Blockchain Framework-Enabled Collaborative Intrusion Detection for Protecting IoT and Cloud Networks. *IEEE Internet of Things Journal*, *8*(12), 9463–9472. doi:10.1109/JIOT.2020.2996590

Alexandre, A. B., Rogério, B. a., Julio, C. R., & Antonio, B. (2018). An exploration of blockchain technology in supply chain. In *22nd Cambridge International Manufacturing Symposium*. University of Cambridge.

Alkhalifah, A., Ng, A., Watters, P., & Kayes, A. (2021). A Mechanism to Detect and Prevent Ethereum Blockchain Smart Contract Reentrancy Attacks. *Frontiers in Computer Science.*

Almakhour, M., Wehby, A., Sliman, L., Samhat, A., & Mellouk, A. (2021). Smart Contract Based Solution for Secure Distributed SDN. *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1-6.

Anastasia, M., & Aaron, L. (2018). Designing secure Ethereum smart contracts: A finite state machine-based approach. *Proceedings of the 22nd International Conference on Financial Cryptography Data Security*, 1–15.

Anastasia, M., & Aaron, L. (2018). FSolidM for Designing Secure Ethereum Smart Contracts. *proceeding of the 7th International Conference on Principles of Security and Trust (POST).*

Anastasia, M., & Aaron, L. (2019). *VeriSolid: Correct-by-Design Smart Contracts for Ethereum.* arXiv:1901.01292v2.

Anton, P., Dimitar, D., Petar, T., Dana, D., & Martin, V. (2020). VerX: Safety Verification of Smart Contracts. *IEEE Symposium on Security and Privacy (SP)*, 1661-1677.

Antonino, P., & Roscoe, A. (2020). *Formalising and verifying smart contracts with Solidifier: a bounded model checker for Solidity.* arXiv, abs/2002.02710.

Aquilina, S.J., Casino, F., Vella, M., Ellul, J., & Patsakis, C. (2021). *EtherClue: Digital investigation of attacks on Ethereum smart contracts.* ArXiv,abs/2104.05293.

Ari, J., Ahmed, K., & Elaine, S. (2016). The Ring of Gyges: Investigating the Future of Criminal Smart Contracts. In *SIGSAC Conference on Computer and Communications Security* (pp. 283-295). ACM.

Ashizawa, N., Yanai, N., Cruz, J. P., & Okamura, S. (2021). Eth2Vec: Learning Contract-Wide Code Representations for Vulnerability Detection on Ethereum Smart Contracts. *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. doi:10.1145/3457337.3457841

Ashraf, I., Ma, X., Jiang, B., & Chan, W. K. (2020). GasFuzzer: Fuzzing Ethereum Smart Contract Binaries to Expose Gas-Oriented Exception Security Vulnerabilities. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 99552–99564. doi:10.1109/ACCESS.2020.2995183

Babel, K., Daian, P., Kelkar, M., & Juels, A. (2021). *Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts.* Academic Press.

Barati, M., Buchanan, W.J., Lo, O., & Rana, O.F. (2021). *A Privacy-Preserving Platform for Recording COVID-19 Vaccine Passports.* ArXiv, abs/2112.01815.

Barboni, M., Morichetta, A., & Polini, A. (2021). SuMo: A Mutation Testing Strategy for Solidity Smart Contracts. *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, 50-59. doi:10.1109/AST52587.2021.00014

Bartoletti, M., & Pompianu, L. (2018). *An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns.* arXiv:1703.06322.

Benedikt, B., Shashank, A., Mahdi, Z., & Dan, B. (2019). *Zether: Towards Privacy in a Smart Contract World.* International Association for Cryptologic Research (IACR) ePrint Archive.

Bill, M., & Ari, J. (2016). Setting standards for altering and undoing smart contracts. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web* (pp. 151-166). Springer.

Boron, M., & Kobusińska, A. (2021). Alternative Authentication with Smart Contracts for Online Games. *2021 IEEE 46th Conference on Local Computer Networks (LCN),* 415-418.

Brent, L., Jurisevic, A., Kong, M., Liu, E., Gauthier, F., Gramoli, V., Holz, R., & Scholz, B. (2018). *Vandal: A Scalable Security Analysis Framework for Smart Contracts.* arXiv abs/1809.03981.

Çagdas, Iguernlala, Laporte, Fessant, & Mebsout. (2018). Liquidity: Ocaml pour la blockchain. *Journées Francophones des Langages Applicatifs.*

Chao, L., Han, L., Zhao, C., Zhong, C., Bangdao, C., & Bill, R. (2018). Reguard: finding reentrancy bugs in smart contracts. *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 65–68.

Chao, P., Sefa, A., & Ajitha, R. (2019). *A Framework for Solidity Contract Instrumentation and Analysis.* arXiv:1905.01659v1.

Cheng, R., Zhang, F., Kos, J., He, W., Hynes, N., Johnson, N. M., Juels, A., Miller, A. K., & Song, D. (2019). Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts. *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, 185-200. doi:10.1109/EuroSP.2019.00023

Chibuzor, U., Aleksandr, K., Kondwani, T., & Alex, N. (2018). *An Exploration of Blockchain enabled Smart-Contracts Application in the Enterprise*. Technical report.

Chinen, Y., Yanai, N., Cruz, J., & Okamura, S. (2020). *Hunting for Re-Entrancy Attacks in Ethereum Smart Contracts via Static Analysis.* arXiv, abs/2007.01029.

Christof, F. T., & Julian, S. (2018). Osiris: Hunting for Integer Bugs in Ethereum Smart Contracts. *2018 Annual Computer Security Applications Conference (ACSAC '18)*. doi:10.1145/3274694.3274737

Christof, F. T., & Mathis, S. (2019). *The art of the scam: Demystifying honeypots in ethereum smart contracts.* arXiv:1902.06976.

Christopher, K. F., & Mariusz, N. (2016). From institutions to code: Towards automated generation of smart contracts. IEEE 1st International Workshops on Foundations and Applications of Self Systems (FAS*W), 210-215.

Christopher, N., & Vincent, G. (2016). The blockchain anomaly. In *15th International Symposium on Network and Computer Applications* (pp. 301-317). IEEE.

Coblenz, M. (2017). Obsidian: A safer blockchain programming language. *Proc. IEEE/ACM 39th International Conference on Software Engineering Companion,* 1–11. doi:10.1109/ICSE-C.2017.150

Contro, F., Crosara, M., Ceccato, M., & Preda, M. D. (2021). EtherSolve: Computing an Accurate Control-Flow Graph from Ethereum Bytecode. *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, 127-137.

Crypto, P. (2018). *The $3 Million Winner of Fomo3D Is Still Playing to Win*. Retrieved from https://www.longhash.com/en/news/2257/The-$3-Million-Winner-of-Fomo3D-Is-Still-Playing-to-Win

Cyril, G., & Ricardo, P. (2019). *Selfish Mining in ethereum*. arXiv:1904.13330.

Damle, S., Gujar, S., & Moti, M. H. (2021). FASTEN: Fair and Secure Distributed Voting Using Smart Contracts. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1-3. doi:10.1109/ICBC51069.2021.9461060

Daniel, P., & Benjamin, L. (2019). *Smart contract vulnerabilities: Does anyone care?* arXiv:1902.06710v2.

David, C. S. (2020). *Private and Verifiable Smart Contracts on Blockchains.* arXiv:1807.09484v2.

de Pedro, Daniele, & Cuende. (2017). *Witnet: A decentralized oracle network protocol.* arXiv:1711.09756.

Dietrich, F., Turgut, A., Palm, D., & Louw, L. (2020). *Smart Contract-Based Blockchain Solution to Reduce Supply Chain Risks*. APMS.

Ding, M., Li, P., Li, S., & Zhang, H. (2021). *HFContractFuzzer: Fuzzing Hyperledger Fabric Smart Contracts for Vulnerability Detection*. Evaluation and Assessment in Software Engineering.

DocumentationP. (n.d.). Available at: https://docs.provable.xyz

Egala, B. S., Pradhan, A., Badarla, V., & Mohanty, S. (2021). Fortified-Chain: A Blockchain-Based Framework for Security and Privacy-Assured Internet of Medical Things With Effective Access Control. *IEEE Internet of Things Journal*, *8*(14), 11717–11731. doi:10.1109/JIOT.2021.3058946

El Majdoubi, D., El Bakkali, H., & Sadki, S. (2021). SmartMedChain: A Blockchain-Based Privacy-Preserving Smart Healthcare Framework. *Journal of Healthcare Engineering*, *2021*, 1–19. doi:10.1155/2021/4145512 PMID:34777733

Ellis, S., Juels, A., & Nazarov, S. (2017). *ChainLink: A decentralized oracle network*. White Paper. Available:https://link.smartcontract.com/whitepaper

Elvira, A., Jesus, C., Pablo, G., Guillermo, R., & Albert R. (2019). *SAFEVM: A Safety Verifier for Ethereum Smart Contract*. arXiv:1906.04984v1

Elvira, A., Jesus, C., Pablo, G., Guillermo, R., & Albert, R. (2019). *GASOL: Gas Analysis and Optimization for Ethereum Smart Contracts,* arXiv:1912.11929.

Eshghie, M., Artho, C., & Gurov, D. (2021). *Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning*. Evaluation and Assessment in Software Engineering. doi:10.1145/3463274.3463348

Eskandari, Clark, Sundaresan, & Adham. (2017). On the feasibility of decentralized derivatives markets. *Proc. Int. Conf. Financial Cryptography Data Secur.*, 553–567. doi:10.1007/978-3-319-70278-0_35

Evgeniy, S. (2018). *Debugging Smart Contract's Business Logic Using Symbolic Model Checking*. arXiv:1812.00619v1.

Eyal, I., & Sirer, E. G. (2014). *How to disincentivize large bitcoin mining pools*. https://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/

Fan, S., Fu, S., Xu, H., & Cheng, X. (2021). Al-SPSD: Anti-leakage smart Ponzi schemes detection in blockchain. *Information Processing & Management*, *58*(4), 102587. doi:10.1016/j.ipm.2021.102587

Fan, Z., Ethan, C., Kyle, C., Ari, J., & Elaine, S. (2016). Town crier: An authenticated data feed for smart contracts. In *ACM SIGSAC Conference on Computer and Communications Security* (pp. 270-282). ACM.

Fang, M., Zhang, Z., Jin, C., & Zhou, A. (2021). High-Performance Smart Contracts Concurrent Execution for Permissioned Blockchain Using SGX. *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 1907-1912.

Florian, I., Guido, G., Regis, R., & Giovanni, S. (2016). Evaluation of logic-based smart contracts for blockchain systems. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web* (pp. 167-183). Springer.

Fox, P. (2021). TrustedAP: Using the Ethereum Blockchain to Mitigate the Evil Twin Attack. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. doi:10.1145/3408877.3439695

Frank, J., Aschermann, C., & Holz, T. (2020). ETHBMC: A Bounded Model Checker for Smart Contracts. *Proceeding of the 29th USENIX Security Symposium*.

Franklin, S., Daniel, H., Alexander, H., Sophia, D., & Susan, E. (2019). *Flint for Safer Smart Contracts.* arXiv:1904.06534.

Gallersdörfer, U., & Matthes, F. (2021). TeSC: TLS/SSL-Certificate Endorsed Smart Contracts. *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 95-100. doi:10.1109/DAPPS52256.2021.00016

Gao, J., Liu, H., Liu, C., Li, Q., Guan, Z., & Chen, Z. (2019). EASYFLOW: Keep Ethereum Away from Overflow. *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 23-26.

Garfatta, I., Klai, K., Gaaloul, W., & Graiet, M. (2021). A Survey on Formal Verification for Solidity Smart Contracts. *2021 Australasian Computer Science Week Multiconference.*

Geng, Z., He, Y., Wang, C., Xu, G., Xiao, K., & Yu, S. (2021). A Blockchain based Privacy-Preserving Reputation Scheme for Cloud Service. *ICC 2021 - IEEE International Conference on Communications*, 1-6.

Grossman, Abraham, Golan-Gueta, Michalevsky, Rinetzky, Sagiv, & Zohar. (2017). Online detection of effectively callback free objects with applications to smart contracts. *Proceedings of the ACM on Programming Languages*.

Gustavo, G., Will, S., Artur, C., Josselin, F., & Alex, G. (2020). Echidna: Effective, Usable, and Fast Fuzzing for Smart Contracts. *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '20)*.

Hamda, A., Muhammad, H. R., Khaled, S., & Davor, S. (2020). Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges. *IEEE Access: Practical Innovations, Open Solutions*.

Han, Liu, Zhao, Jiang, & Sun.(2018). S-gram: Towards Semantic-Aware Security Auditing for Ethereum Smart Contracts. *33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 814-819. doi:10.1145/3238147.3240728

Hess, Malahov, & Petterson. (2017). *Aeternity Blockchain*. Available:https://aeternity.com/aeternityblockchainwhitepaper.pdf

Hiroki, W., Shigeru, F., Atsushi, N., Yasuhiko, M., & Akihito, A. (2015). Blockchain contract: A complete consensus using Blockchain. *4th Global Conference on Consumer Electronics (GCCE)*, 577-578.

Houda, Z. A., Hafid, A., & Khoukhi, L. (2019). Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 98893–98907. doi:10.1109/ACCESS.2019.2930715

Hu, B., Zhang, Z., Liu, J., Liu, Y., Yin, J., Lu, R., & Lin, X. (2021). *A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems*. Patterns.

Hu, X., Zhuang, Y., Lin, S., Zhang, F., Kan, S., & Cao, Z. (2021). A security type verifier for smart contracts. *Computers & Security*, *108*, 102343. doi:10.1016/j.cose.2021.102343

Huashan, C., Marcus, P., Laurent, N., & Shouhuai, X., (2020). A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. *ACM Comput. Surv., 53*. . 10.1145/3391195

Ioannis, K., Maria, P., & Nedaa, B. A. (2018). Design of the Blockchain Smart Contract: A Use case for real estate. Journal of Information Security, 177-190.

Ittay, E., & Emin, S. (2018). Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, *61*(7), 95–102. doi:10.1145/3212998

Ivanov, N., Yan, Q., & Wang, Q. (2021). *Blockumulus: A Scalable Framework for Smart Contracts on the Cloud.* ArXiv, abs/2107.04904.

Ivica, N., Aashish, K., Ilya, S., Prateek, S., & Aquinas, H. (2018). Finding The Greedy, Prodigal, and Suicidal Contracts at Scale. *34th Annual Computer Security Applications Conference*, 653-663.

Jiachi, C. (2020). Finding Ethereum Smart Contracts Security Issues by comparing history. *35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20)*.

Jialiang, C., Bo, G., Hao, X., Jun, S., Yan, C., & Zijiang, Y. (2019). *sCompile: Critical Path Identification and Analysis for Smart contract.* arXiv:1808.00624v2

Jiang, B., Liu, Y., & Chan, W. K. (2018). ContractFuzzer: Fuzzing Smart Contracts for Vulnerability Detection. *Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE'18)*. doi:10.1145/3238147.3238177

Jin, C., Pang, S., Qi, X., Zhang, Z., & Zhou, A. (2021). A High Performance Concurrency Protocol for Smart Contracts of Permissioned Blockchain. *IEEE Transactions on Knowledge and Data Engineering*, 1. doi:10.1109/TKDE.2021.3059959

Jing, L., & Zhentian, L. (2019). *A Survey on Security Verification of Blockchain Smart Contracts* (Vol. 7). IEEE Access. doi:10.1109/ACCESS.2019.2921624

Jingxuan, H., Mislav, B., Nodar, A., Petar, T., & Martin, V. (2019). Learning to Fuzz from Symbolic Execution with Application to Smart Contracts. *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. doi:10.1145/3319535.3363230

Johannes, K., & Christian, R. (2018). Teether: Gnawing at Ethereum to Automatically Exploit Smart Contracts. *Proceedings of the 27th USENIX Security Symposium*.

Josselin, F., Gustavo, G., & Alex, G. (2019). Slither: A Static Analysis Framework For Smart Contract. *Proceedings IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. doi:10.1109/WETSEB.2019.00008

Juan, G., & Pawel, S. (2018). *PDFS: Practical Data Feed Service for Smart Contracts.* arXiv 1808.06641v2.

Kamboj, P., Khare, S., & Pal, S. (2021). User authentication using Blockchain based smart contract in role-based access control. *Peer-to-Peer Networking and Applications*, *14*(5), 2961–2976. doi:10.1007/s12083-021-01150-1

Karl, W., Sinisa, M., Silvan, E., Kari, K., & Srdjan, C. (2019). ACE: Asynchronous and Concurrent Execution of Complex Smart Contracts. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security.*

Kevin, D., Mitchell, A., Ahmed, K., Andrew, M., & Elaine, S. (2016). Step by Step Towards Creating a Safe Smart Contract: Lessons and insights from a cryptocurrency lab. In *International Conference on Financial Cryptography and data Security* (pp. 79-84). Springer.

Khan, J. A., Bangalore, K. U., Kurkcu, A., & Ozbay, K. (2021). TREAD: Privacy Preserving Incentivized Connected Vehicle Mobility Data Storage on InterPlanetary-File-System-Enabled Blockchain. *Transportation Research Record: Journal of the Transportation Research Board*.

Khan, S., Amin, M. B., Azar, A. T., & Aslam, S. (2021). Towards Interoperable Blockchains: A Survey on the Role of Smart Contracts in Blockchain Interoperability. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 116672–116691. doi:10.1109/ACCESS.2021.3106384

Khan, S. N., Loukil, F., Ghedira, C., Benkhelifa, E., & Bani-Hani, A. I. (2021). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Networking and Applications*, *14*(5), 1–25. doi:10.1007/s12083-021-01127-0 PMID:33897937

Kim & Kim. (2020). Intelligent Mediator-based Enhanced Smart Contract for Privacy Protection. *ACM Trans. Internet Technol., 21*(1). 10.1145/3404892

Knecht, M., & Stiller, B. (2021). SAMOS: a Smart Contract Access Management over Opaque and Substructural Types. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1-5. doi:10.1109/ICBC51069.2021.9461117

Konrad, W., & Julian, S. (2019). *Annotary: A Concolic Execution System for Developing Secure Smart Contracts.* ESORICS.

Konstantinos, C., & Michael, D. (2016). Blockchains and Smart Contracts for IOT. IEEE special section on the Plethora of Research in Internet of Things (iot), 2292-2303.

Li, C., Qu, X., & Guo, Y. (2021). TFCrowd: A blockchain-based crowdsourcing framework with enhanced trustworthiness and fairness. *EURASIP Journal on Wireless Communications and Networking*, *2021*, 1–20. doi:10.1186/s13638-020-01861-8

Li, R., & Asaeda, H. (2021). FSCC: Flexible Smart Contract Interaction with Access Control for Blockchain. *ICC 2021 - IEEE International Conference on Communications*, 1-6.

Liu, B., Sun, S., & Szalachowski, P. (2020). SMACS: Smart Contract Access Control Service. 2020. *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 221-232. doi:10.1109/DSN48063.2020.00039

Liu, Z., Qian, P., Wang, X., Zhuang, Y., Qiu, L., & Wang, X. (2021). *Combining Graph Neural Networks with Expert Knowledge for Smart Contract Vulnerability Detection.* ArXiv, abs/2107.11598.

Loi, Teutsch, Kulkarni, & Saxena. (2015). Demystifying incentives in the consensus computer. *Proceedings of the ACM CCS*. 706–719.

Loi, L., Duc-Hiep, C., Hrishi, O., Prateek, S., & Aquinas, H. (2016). Making Smart Contracts Smarter. *ACM SIGSAC Conference on Computer and Communications Security*, 167-183.

Lutz, O., Chen, H., Fereidooni, H., Sendner, C., Dmitrienko, A., Sadeghi, A., & Koushanfar, F. (2021). *ESCORT: Ethereum Smart COntRacTs Vulnerability Detection using Deep Neural Network and Transfer Learning.* ArXiv, abs/2103.12607.

Maher, A., & Aad Van, M. (2017). Blockchain-based Smart Contracts: A systematic mapping study. *3rd International Conference on Artificial Intelligence and Soft Computing*, 131-135.

Marc, J., Farouk, H., Ramy, G., & Ziyaad, Q. (2019). *Do Smart Contract Languages Need to be Turing Complete?* arXiv:1710.06372.

Maria, A., Aviv, Z., & Laurent, V. (2017). Hijacking Bitcoin: Routing attacks on cryptocurrencies. *Security and Privacy (SP), IEEE Symposium on*, 375–392.

Maria, A., Gian, M., Jan, M., & Laurent, V. (2018). *SABRE: Protecting Bitcoin against Routing Attacks.* arXiv:1808.06254.

Mark, M., Felipe, M., Eric, H., Alex, G., Gustavo, G., Josselin, F., Trent, B., & Artem, D. (2019). *Manticore: A User-Friendly Symbolic Execution Framework for Binaries and Smart Contracts.* arXiv:1907.03890v3.

Markus, K. (2019). *Mandala: A smart Contract Programming*. arXiv:1911.11376v1.

Mars, R., Abid, A., Cheikhrouhou, S., & Kallel, S. (2021). A Machine Learning Approach for Gas Price Prediction in Ethereum Blockchain. *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 156-165.

Massimo, B., Barbara, P., & Sergio, S. (2018). Data mining for detecting bitcoin ponzi schemes. *Crypto Valley Conference on Blockchain Technology (CVCBT)*, 75–84.

Massimo, B., Letterio, G., & Maurizio, M. (2020). *A true concurrent model of smart contracts executions.* arXiv 1905.04366v3.

Mazurok, I., Leonchyk, Y., Antonenko, O., & Volkov, K. S. (2021). *Smart contract sharding with proof of execution*. Applied Aspects of Information Technology. doi:10.15276/aait.03.2021.6

Mengya, Z., Xiaokuan, Z., Yinqian, Z., & Zhiqiang, L. (2020), TxSpector: Uncovering Attacks in Ethereum from Transactions. *Proc. 29th USENIX Security Symposium*.

Mi, F., Wang, Z., Zhao, C., Guo, J., Ahmed, F., & Khan, L. (2021). VSCL: Automating Vulnerability Detection in Smart Contracts with Deep Learning. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1-9. doi:10.1109/ICBC51069.2021.9461050

Michael, R., Wenting, L., Ghassan, O. K., & Lucas, D. (2019). Sereum: Protecting Existing Smart Contracts Against Re-entrancy attacks. *Network and Distributed Systems Security (NDSS) Symposium 2019*.

Miller, A., Kosba, A., Katz, J., & Shi, E. (2015). Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. *Proceedings of the ACM CCS*, 680–691. doi:10.1145/2810103.2813621

Monika, D., & Gernot, S. (2019). *A Survey of Tools for analysing Smart contract*. IEEE.

Muchhala, Y., Singhania, H., Sheth, S., & Devadkar, K. (2021). Enabling MapReduce based Parallel Computation in Smart Contracts. *2021 6th International Conference on Inventive Computation Technologies (ICICT),* 537-543.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Retrieved from: https://bitcoin.org/bitcoin.pdf

Negara, E. S., Hidayanto, A. N., Andryani, R., & Syaputra, R. (2021). Survey of Smart Contract Framework and Its Application. *Inf.*, *12*(7), 257. doi:10.3390/info12070257

Neville, G., Michael, K., Anton, J., Lexi, B., & Bernhard, S. (2018). MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts. *Proc. ACM Program. Lang. 2, OOPSLA*. doi:10.1145/3276486

Nguyen, T. D., Pham, L. H., & Sun, J. (2021). SGUARD: Towards Fixing Vulnerable Smart Contracts Automatically. *2021 IEEE Symposium on Security and Privacy (SP)*, 1215-1229. doi:10.1109/SP40001.2021.00057

O'Connor, R. (2017). Simplicity: A new language for blockchains. *Proc Workshop Program. Lang. Anal. Secur.,* 107–120.

Omar, A., Jamil, A. K., Khandakar, A., Uzzal, A. R., Bosri, R., Mansoor, N., & Rahman, M. S. (2021). A Transparent and Privacy-Preserving Healthcare Platform With Novel Smart Contract for Smart Cities. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 90738–90749. doi:10.1109/ACCESS.2021.3089601

Omar, I. A., Jayaraman, R., Debe, M., Salah, K., Yaqoob, I., & Omar, M. A. (2021). Automating Procurement Contracts in the Healthcare Supply Chain Using Blockchain Smart Contracts. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 37397–37409. doi:10.1109/ACCESS.2021.3062471

Peng, K., Li, M., Huang, H., Wang, C., Wan, S., & Choo, K. R. (2021). Security Challenges and Opportunities for Smart Contracts in Internet of Things: A Survey. *IEEE Internet of Things Journal*, *8*(15), 12004–12020. doi:10.1109/JIOT.2021.3074544

Peng, Q., Zhenguang, L., Qinming, H., Roger, Z., & Xun, W. (2020). Towards Automated Reentrancy Detection for Smart Contracts Based on Sequential Models. *IEEE Access: Practical Innovations, Open Solutions*.

Petar, T., Andrei, D., Dana, D-C., Arthur, G., Florian, B., & Martin, V. (2018). Securify: Practical Security Analysis of Smart Contracts. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 67-82. doi:10.1145/3243734.3243780

Peterson, K. Williams, & Alexander. (2015). *Augur: A decentralized oracle and prediction market platform.* arXiv:1501.01042.

Pettersson, J., & Edström, R. (2018). *Safer Smart Contracts Through TypeDriven Development*. Available:https://publications.lib.chalmers.se/records/fulltext/234939/234939.pdf

Pierro, G. A. (2021). Smart-Graph: Graphical Representations for Smart Contract on the Ethereum Blockchain. *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 708-714. doi:10.1109/SANER50967.2021.00090

Pradhan, N., & Singh, A. (2021). Smart contracts for automated control system in Blockchain based smart cities. *Journal of Ambient Intelligence and Smart Environments*, *13*(3), 253–267. doi:10.3233/AIS-210601

Bernabe, Canovas, Hernandez-Ramos, Moreno, & Antonio. (2019). Privacy-Preserving Solutions for Blockchain: Review and Challenges. *IEEE Access: Practical Innovations, Open Solutions*.

Qasse, I. A., Mishra, S., & Hamdaqa, M. (2021). iContractBot: A Chatbot for Smart Contracts' Specification and Code Generation. *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, 35-38. doi:10.1109/BotSE52550.2021.00015

Qin, P., Tan, W., Guo, J., & Shen, B. (2021). Intelligible Description Language Contract (IDLC) – A Novel Smart Contract Model. *Information Systems Frontiers*, 1–18. doi:10.1007/s10796-021-10138-4

Quan, N., Andre, C., & Michael, K. (2020), *OV: Validity-Based Optimistic Smart Contracts.* arXiv:2004.04338v1.

RachelR. (2018). Retrieved from https://www.coindesk.com/four-projects-seek

Rahman, M. U., Baiardi, F., Guidi, B., & Ricci, L. (2019) Protecting Personal Data Using Smart Contracts. In *Internet and Distributed Computing Systems. IDCS 2019. Lecture Notes in Computer Science, vol 11874*. Springer. 34914-1_310.1007/978-3-030-

Rajesh, G., Sudeep, T., Fadi, A., Prit, I., Ali, N., & Sung, W. K. (2020). Smart Contract Privacy Protection Using AI in Cyber-Physical Systems: Tools, Techniques and Challenges. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 24746–24772. doi:10.1109/ACCESS.2020.2970576

Regnath, E., & Steinhorst, S. (2018). SmaCoNat: Smart contracts in natural language. *Proc. Forum Specification Design Lang*., 5–16. doi: doi:10.1109/FDL.2018.8524068

Ren, Q., Liu, H., Li, Y., & Lei, H. (2021). *CLOAK: A Framework For Development of Confidential Blockchain Smart Contracts.* ArXiv, abs/2106.13460.

Reza, M. P., Ali, D., Kim-Kwang, R. C., & Amritraj, S. (2018). Empirical Vulnerability Analysis of Automated Smart Contracts Security Testing on Blockchains. *Proceedings of 28th Annual International Conference on Computer Science and Software Engineering*.

Rodler, M., Li, W., Karame, G., & Davi, L. (2020). *EVMPatch: Timely and Automated Patching of Ethereum Smart Contracts.* arXiv abs/2010.00341.

Samreen, N. F., & Alalfi, M. (2020). Reentrancy Vulnerability Identification in Ethereum Smart Contracts. *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 22-29.

Samreen, N.F., & Alalfi, M. (2021). *A Survey of Security Vulnerabilities in Ethereum Smart Contracts.* ArXiv, abs/2105.06974.

Samreen, N. F., & Alalfi, M. (2021). SmartScan: An approach to detect Denial of Service Vulnerability in Ethereum Smart Contracts. *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB),* 17-26.

Santiago, B., Henrique, R., Marcus, D., & Stéphane, D. (2018). SmartInspect: SolidparticSmart Contract Inspector. *IWBOSE 2018 - 1st International Workshop on Blockchain Oriented Soft-ware Engineering*. hal-01831075 doi:10.1109/IWBOSE.2018.8327566

Saquib, N., Bakir, F., Krintz, C., & Wolski, R. (2021). *A Resource-Efficient Smart Contract for Privacy Preserving Smart Home Systems*. Academic Press.

Sara, R. & Ralph, D. (2019). Security, Performance, and Applications of Smart Contracts: A Systematic Survey. *IEEE Access.*

Sarwaar, S., Hector, M., & Tom, C. (2020). *Smart Contract: Attacks and Protections. IEEE Access*. doi:10.1109/Access.2020.2970495

Schrans, F., Eisenbach, S., & Drossopoulou, S. (2018), Writing safe smart contracts in flint. *Proc. Conf. Companion 2nd Int. Conf. Art, Sci., Eng. Program*, 218–219.

Seike, H., Aoki, Y., & Koshizuka, N. (2021). Towards Smart Contracts for Verifying DNN Model Generation Process with the Blockchain. *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, 160-168.

Sergei, T., Ekaterina, V., Ivan, I., Ramil, T., Evgeny, M., & Yaroslav, A. (2018). SmartCheck: StaticAnalysis of Ethereum Smart Contracts. In EEE/ACM1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB 2018). ACM. https://doi.org/10.1145/3194113.3194115.

Shuai, W., Liwei, O., Yong, Y., Xiaochun, N., Xuan, H., & Fei-Yue, W. (2019). Blockchain-Enabled Smart Contracts: Architecture, applications and future trends. *IEEE Transactions on Systems, Man, and Cybernetics. Systems*, 2168–2216.

Solaiman, E., Wike, T., & Sfyrakis, I. (2021). Implementation and evaluation of smart contracts using a hybrid on- and off-blockchain architecture. *Concurrency and Computation*.

Solomon, R., & Almashaqbeh, G. (2021). SmartFHE: Privacy-Preserving Smart Contracts from Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch*.

Song, L., Zhu, Z., Li, M., Ma, L., & Ju, X. (2021). A Novel Access Control for Internet of Things Based on Blockchain Smart Contract. *2021 IEEE 5th Advanced Information Technology*, *Electronic and Automation Control Conference (IAEAC)*, 5, 111-117.

Stephens, J., Ferles, K., Mariano, B., Lahiri, S. K., & Dillig, I. (2021). SmartPulse: Automated Checking of Temporal Properties in Smart Contracts. *2021 IEEE Symposium on Security and Privacy (SP)*, 555-571.

Sukrit, K., Seep, G., Mohan, D., & Subodh, S. (2018). ZEUS: Analyzing Safety of Smart Contracts. *Network and Distributed Systems Security (NDSS) Symposium,* 1-15.

Sunbeom, S., Myungho, L., Jisu, P., Heejo, L., & Hakjoo, O. (2019). *VERISMART: A Highly Precise Safety Verifier for Ethereum Smart Contracts.* arXiv:1908.11227v2.

Suvitha, M., & Subha, R. (2021). A Survey on Smart Contract Platforms and Features. *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 1*, 1536-1539.

Tang, X., Zhou, K., Cheng, J., Li, H., & Yuan, Y. (2021). The Vulnerabilities in Smart Contracts: A Survey. *Advances in Artificial Intelligence and Security.*

Tesnim, A., & Kei-Leo, B. (2018). Formal Verification of Smart Contracts Based on Users and Blockchain Behaviors Models. *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS).*

Thomas, D., Paul, G., Maurice, H., & Eric, K. (2017). *Adding Concurrency to Smart Contracts.* arXiv:1702.04467v1.

Thomas, G., Thomas, J., & Justine, S. (2020). *Termination of Ethereum's Smart Contracts* [Research Report]. Univ Rennes, Inria, CNRS, IRISA.

Tianyu, S., & Wensheng, Y. (2020). A Formal Verification Framework for Security Issues of Blockchain Smart Contracts. *IEEE Transactions on Emerging Topics in Computing*.

Timuçin, T., & Birogul, S. (2021). A survey: Making "Smart Contracts" really smart. *Transactions on Emerging Telecommunications Technologies*.

Ting, C., Xiaoqi, L., Xiapu, L., & Xiaosong, Z. (2017). Under-optimized smart contracts devour your money. In *24th International Conference on Software Analysis, Evolution and Reengineering* (pp. 442-446). IEEE.

Ting, C., Youzheng, F., Zihao, L., Hao, Z., Xiapu, L., Xiaoqi, L., Xiuzhuo, X., Jiachi, C., & Xiaosong, Z. (2020). GasChecker: Scalable Analysis for Discovering Gas-Inefficient Smart Contracts. *IEEE Transactions on Emerging Topics in Computing*. Advance online publication. doi:10.1109/TETC.2020.2979019

Ting, C., Zihao, L., Hao, Z., Jiachi, C., Xiapu, L., Xiaoqi, L., & Xiaosong, Z. (2018). Towards Saving Money in Using Smart Contracts. In ICSE-NIER'18: 40th International Conference on Software Engineering: New Ideas and Emerging Results Track, May 27-June 3, 2018. ACM. https://doi.org/10.1145/3183399.3183420.

Tjiam, K., Wang, R., Chen, H., & Liang, K. 2021. Your Smart Contracts Are Not Secure: Investigating Arbitrageurs and Oracle Manipulators in Ethereum. *Proceedings of the 3rd Workshop on Cyber-Security Arms Race (CYSARM '21).* doi:10.1145/3474374.3486916

Torres, C. F., Baden, M., Norvill, R., Pontiveros, B. B., Jonker, H., & Mauw, S. (2020). *AEGIS: Shielding Vulnerable Smart Contracts Against Attacks.* arXiv:2003.05987v1.

Torres, C.F., Jonker, H., & State, R. (2021). *Elysium: Automagically Healing Vulnerable Smart Contracts Using Context-Aware Patching.* ArXiv, abs/2108.10071.

Tyurin, A. V., Tyuluandin, I. V., Maltsev, V. S., Kirilenko, I. A., & Berezun, D. A. (2019). Overview of the Languages for Safe Smart Contract Programming. *Trudy ISP RAN/Proc. ISP RAS, 31*(3), 157-176. DOI: doi:10.15514/ISPRAS- 2019-31(3)-13

Vangala, A., Sutrala, A. K., Das, A., & Jo, M. (2021). Smart Contract-Based Blockchain-Envisioned Authentication Scheme for Smart Farming. *IEEE Internet of Things Journal*, *8*, 10792–10806.

Vardhini, B., Dass, S. N., Sahana, R., & Chinnaiyan, R. (2021). A Blockchain based Electronic Medical Health Records Framework using Smart Contracts. *International Conference on Computer Communication and Informatics (ICCCI)*, 1-4.

Wang, B., Liu, H., Liu, C., Yang, Z., Ren, Q., Zheng, H., & Lei, H. (2021). BLOCKEYE: Hunting for DeFi Attacks on Blockchain. *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 17-20.

Wang, S., Yuan, Y., Wang, X., Li, J., Qin, R., & Wang, F. (2018). An Overview of Smart Contract: Architecture, Applications, and Future Trends. *2018 IEEE Intelligent Vehicles Symposium (IV)*, 108-113.

Wang, W., Song, J., Xu, G., Li, Y., Wang, H., & Su, C. (2020). ContractWard: Automated Vulnerability Detection Models for Ethereum Smart Contracts. *IEEE Transactions on Network Science and Engineering*.

Wang, X., He, J., Xie, Z., Zhao, G., & Cheung, S. (2020). ContractGuard: Defend Ethereum Smart Contracts with Embedded Intrusion Detection. *IEEE Transactions on Services Computing*, *13*, 314–328.

Wang, Y., He, J., Zhu, N., Yi, Y., Zhang, Q., Song, H., & Xue, R. (2021). Security enhancement technologies for smart contracts in the blockchain: A survey. *Transactions on Emerging Telecommunications Technologies*.

Wei, Y., Kan, L., Yi, D., Guang, Y., & Kai, H. (2018). A Parallel Smart Contract Model. *Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence,* 72-77.

Weili, C., Zibin, Z., Jiahui, C., Edith, N., Peilin, Z., & Yuren, Z. (2018). Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology. *WWW 2018: The 2018 Web Conference*. 10.1145/3178876.3186046

Wen, X., Guan, Z., Li, D., Lyu, H., & Li, H. (2021). A Blockchain-based Framework for Information Management in Internet of Vehicles. *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 18-23.

Wenbo, W., Dinh, T. H., Peizhao, H., Zehui, X., Dusit, N., Ping, W., Yonggang, W., & Dong, I. (2019). A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 22328–22370.

Wesley, J. T., Xing, J. H., Sourav, S. G., & Yew-Soon, O. (2019). *Towards Safer Smart Contracts: A Sequence Learning Approach to detecting security threats*. arXiv:1811.06632v2.

Wu, L., Wu, S., Zhou, Y., Li, R., Wang, Z., Luo, X., Wang, C., & Ren, K. (2020). *EthScope: A Transaction-centric Security Analytics Framework to Detect Malicious Smart Contracts on Ethereum.* aXiv abs/2005.08278.

Xiao, Qin, Zhu, Chen, & Liu. (2018). SPESC: A Specification Language for Smart Contracts. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, 132-137. doi: doi:10.1109/COMPSAC.2018.00025

Xiao, L. Y., Omar, A., David, L., & Abhik, R. (2020). Smart Contract Repair. *ACM Trans. Softw. Eng. Methodol., 1*. 10.1145/nnnnnnn.nnnnnnn

Xu, Y., Hu, G., You, L., & Cao, C. (2021). *A Novel Machine Learning-Based Analysis Model for Smart Contract Vulnerability*. 10.1155/2021/5798033

Yang, X., Gao, Q., Liu, K., & Gu, H. (2021). Smart contracts based supply chain resource management system in the industrial internet. *2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 31-36.

Yang, X., Ning, Z., Jin, L., Wenjing, L., & Thomas, Y. (2020). *PrivacyGuard: Enforcing Private Data Usage Control with Blockchain and Attested Off-chain Contract*. arXiv:1904 07275.

Yi, Z., Deepak, K., Surya, B., Joshua, M., Andrew, M., & Michael, B. (2018). Erays: Reverse Engineering Ethereum's Opaque Smart Contracts. *Proceedings of the 27th USENIX Security Symposium*.

Yu, F., Emina, T., & Rastislav, B. (2019, February 16). *Precise Attack Synthesis for Smart Contracts.* arXiv:1902.06067v1.

Yuan, R., Xia, Y. B., Chen, H. B., Zang, B. Y., & Xie, J. (2018). ShadowEth: Private Smart Contract on Public Blockchain. *J. Comput. Sci. Technol.*, *33*, 542–556. https://doi.org/10.1007/s11390-018-1839-y

Yuan, Z., Zhenguang, L., Peng, Q., Qi, L., Xiang, W., & Qinming, H. (2020). Smart Contract Vulnerability Detection Using Graph Neural Networks. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 3283 - 3290.

Yuepeng, W., Shuvendu, K., Shuo, C., Rong, P., Isil, D., Cody, B., & Immad, N. (2019). *Formal Specification and Verification of Smart contracts in Azure Blockchain*. arXiv:1812.08829v2.

Yuyao, Z., Siqi, M., Juanru, L., Kailai, L., Surya, N., & Dawu, G. (2020). SMARTSHIELD: Automatic Smart Contract Protection Made Easy. In *27*th International Conference on Software Analysis, Evolution and Reengineering *(SANER)* (pp. 23-34). IEEE.

Zeinab, N., Pierre, Y. P., & Frederic, D. (2018). Model-Checking of Smart Contracts. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 980-987. doi: doi:10.1109/Cybermatics_2018.2018.00185

Zhaofeng, M., Jialin, M., Ji-hui, W., & Zhiguang, S. (2021). Blockchain-Based Decentralized Authentication Modeling Scheme in Edge and IoT Environment. *IEEE Internet of Things Journal*, *8*, 2116–2123.

Zhou, E., Hua, S., Pi, B., Sun, J., Nomura, Y., Yamashita, K., & Kurihara, H. (2018). Security Assurance for Smart Contract. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1-5.

Ziar, R.A., Irfanullah, S., Khan, W.U., & Salam, A. (2021). *Privacy Preservation for On-Chain Data in the Permission less Blockchain using Symmetric Key Encryption and Smart Contract*. Academic Press.

Zibin, Z., Shaoan, X., Hong-Ning, D., Weili, C., Xiangping, C., Jian, W., & Muhammad, I. (2019). *Overview on Smart Contracts: Challenges, Advances and Platforms*. arXiv:1912.10370v1.

Ziftrcoin: A cryptocurrency to enable commerces. (2014). Available at https://d19y4lldx7po3t. cloudfront.net/assets/docs/ziftrcoin-whitepaper-120614.pdf

Senou Rosaire is a Ph.D. student at the Institute of Mathematics and Physics (IMSP), University of Abomey-Calavi, Benin. He received a bachelor's degree in telecommunications at Ghana Technology University College (2011) and a Master Degree at IMSP (2018) where he majored in networking and information systems. He also became a Cisco Certified Network Associate (2011) and received his Advanced Diploma certificate from the City and Guilds of London Institute (2007) in telecommunications. He has more than five years of professional experience in networking and telecommunications and has worked in companies such as Huawei where he was in charge of conducting surveys for Wi-Fi network implementation; supervising fiber network deployment contractors onsite; and proposing solutions to customers. His research embraces networking and blockchain technologies.

Jules Degila (Ph.D.) is a Professor of Computer Science at the University of Abomey-Calavi, Benin. He is specialized in telecommunication networks and systems architecture, deployment, and operations. He has held different management, executive, and board member positions for Western and African telecommunications companies during the last ten years. A guest speaker at various universities in operations research and telecommunication, he has also advised many companies and governments as a strategist in ICT4D. From 04/2005 to 06/2010, he was Assistant Director of Telecommunications Applications and Technologies for a leading Canadian analog and digital television, high-speed Internet, and telephony services provider. During his career in Canada, he was responsible both for the technological architectures of all telecommunications services and for exploring and developing advanced telecommunication applications and technologies. In addition, he served as a technology strategist for regulatory affairs. He was also a member of different working groups of CableLabs (a Denver, Colorado-based research consortium for cable operators). Jules received a Ph.D. degree in electrical engineering from École Polytechnique de Montréal, Canada, in 2004.