Article

Strengthening the Security of Smart Contracts through the Power of Artificial Intelligence

Moez Krichen [1,2]

[1] Department of Information Technology, Faculty of Computer Science and Information Technology, Al-Baha University, Al-Baha 65528, Saudi Arabia; moez.krichen@redcad.org or moez.krichen@ieee.org
[2] ReDCAD Laboratory, University of Sfax, Sfax 3038, Tunisia

Abstract: Smart contracts (SCs) are digital agreements that execute themselves and are stored on a blockchain. Despite the fact that they offer numerous advantages, such as automation and transparency, they are susceptible to a variety of assaults due to their complexity and lack of standardization. In this paper, we investigate the use of artificial intelligence (AI) to improve SC security. We provide an overview of Smart Contracts (SCs) and blockchain technology, as well as a discussion of possible SC-based attacks. Then, we introduce various AI categories and their applications in cybersecurity, followed by a thorough analysis of how AI can be used to enhance SC security. We also highlight the open questions and future directions of research in this field. Our research demonstrates that AI can provide an effective defense against assaults on SCs and contribute to their security and dependability. This article lays the groundwork for future research in the field of AI for SC security.

Keywords: smart contracts; blockchain; cybersecurity; security strengthening; machine learning; artificial intelligence

Citation: Krichen, M. Strengthening the Security of Smart Contracts through the Power of Artificial Intelligence. *Computers* **2023**, *12*, 107. https://doi.org/10.3390/computers12050107

Academic Editors: Ömer Aslan and Refik Samet

Received: 1 May 2023
Revised: 12 May 2023
Accepted: 16 May 2023
Published: 18 May 2023

Copyright: © 2023 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

By automating complex financial transactions, blockchain-based "smart contracts" (SCs) do away with the need for intermediaries like banks or attorneys. They represent a huge advancement in blockchain technology that has the potential to fundamentally change how business is conducted [1].

Some of the most popular blockchain platforms [2] are Ethereum, Hyperledger Fabric, Corda, EOS, and Tron. For creating SCs, each of these platforms has a unique language. One such example is the contract-oriented programming language Solidity, which is used by Ethereum. Similar to JavaScript, Solidity is designed to be simple to learn and use [3,4]. Hyperledger Fabric is based on chaincode, which can be written in Go, Java, or JavaScript [5–7]. The language used to develop Corda is called Kotlin, which is linked to Java [8,9]. EOS and Tron both use C++ and Solidity for SC development, respectively. Different blockchain platforms employ various SC development programming languages, but they all aim to make it simple to develop secure and effective SCs that can function on the blockchain.

The security of SCs, however, is a crucial concern because it can be jeopardized by a variety of risks, including incorrect code, malicious inputs, and attacks on the blockchain network. Due to security flaws, blockchain platforms run the risk of losing money and losing their trust. Therefore, safeguarding SCs is crucial for the development of blockchain applications [10,11].

Numerous high-profile SC hacking incidents have resulted in significant financial losses or data breaches. Here are some notable examples [12,13]:

- The DAO was a crowdfunding effort on the Ethereum blockchain that raised over $150 million in Ether before it was hacked in 2016. However, in June of 2016, a hacker used a SC flaw to steal more than $50 million worth of Ether. A hard fork in the Ethereum blockchain was necessary to restore the missing money;

- A prominent Ethereum wallet known as Parity was hacked in July of 2017 and more than $30 million's worth of Ether was stolen. The Parity multi-signature wallets were compromised due to a flaw in an SC library;
- KingDice is an open-source Ethereum-based gambling platform that was compromised in 2017. A hacker stole almost $300,000 worth of Ether in August 2017 by taking advantage of a flaw in the SC of the network;
- Binance Smart Chain Exploits—In 2021, various SC exploits occurred on Binance Smart Chain, a blockchain platform developed by the cryptocurrency exchange Binance, leading to the theft of millions of dollars' worth of cryptocurrency. Over $200 million's worth of cryptocurrency was stolen through the hacking of a Venus Protocol SC.

These occurrences stress the significance of keeping SCs safe and of being on the lookout for vulnerabilities and attacks at all times. The current methods of SC security include drawbacks and difficulties that must be overcome. The complexity and difficulty in analysis and verification of SCs is a significant obstacle. There may still be undetected vulnerabilities despite the use of code review and rigorous verification. It might be challenging to ensure the ongoing security of SCs because they are frequently updated and often created by distributed teams. Another difficulty is that SCs often operate on public blockchains that are susceptible to attacks from bad actors. Theft of private keys, 51% assaults, and the use of SC coding flaws are all examples of possible attacks [14–17]. In addition, code review and formal verification, two common traditional techniques in SC security, are both costly and time-consuming, making them impracticable for many engineers [18]. Last but not least, the lack of uniformity in SC creation makes it tough to guarantee the safety of SCs on various platforms [15,19].

In light of these difficulties, there is a growing curiosity about how machine learning and AI may be used to bolster SC security [20,21]. By allowing for the detection of anomalies and unusual activity that may indicate a security breach, these methods may give a more thorough and proactive approach to security [22,23]. By fixing these problems, we can make blockchain technology more reliable and inspire more faith in SCs.

This work makes a contribution to SC security research by investigating the potential of artificial intelligence (AI) in this area. In particular, it offers:

- A description of SCs and any potential security holes;
- A discussion of blockchain's involvement in SC security;
- An overview of artificial intelligence, covering its various subtypes and applications in cybersecurity;
- An examination of the potential advantages of implementing AI for SC security, including its capacity to identify and stop threats;
- A review of unresolved problems and potential future study areas in this field.

Following a brief introduction to SCs and artificial intelligence (Sections 2 and 3), the paper delves into the use of AI for SC security (Section 4). This section examines the many ways of using AI in SC security, including as machine learning and deep learning. The study then examines the possible advantages of utilizing AI for SC security, including in its capacity to detect and prevent threats (Section 5). The findings of this study indicate that AI can provide an effective protection against SC attacks. However, there are still unresolved issues and challenges, such as the need for more data and the possibility of adversarial attacks (Section 6). The paper finishes with a summary of the major findings and a discussion on future research in this area (Section 7).

2. Background on Smart Contracts

2.1. Blockchain and Smart Contracts

Blockchain technology is a distributed ledger that allows secure and transparent transactions to take place without the use of intermediaries such as financial institutions or governments [24–26]. It is made up of a network of nodes that work together to maintain a shared database of transactions [27]. Each node has a copy of the database, and the network uses a consensus process to verify all transactions. This ensures that the database cannot be tampered with and that all transactions are transparent and unchangeable. The concept of SCs is a significant breakthrough of blockchain technology. SCs are self-executing programs that operate on a blockchain to automate complex financial transactions without the use of intermediaries. They are saved on the blockchain and are automatically executed when certain conditions are met. Supply chain management, voting systems, and financial derivatives are just a few of the applications for SCs. The general architecture of blockchain is shown in Figure 1.

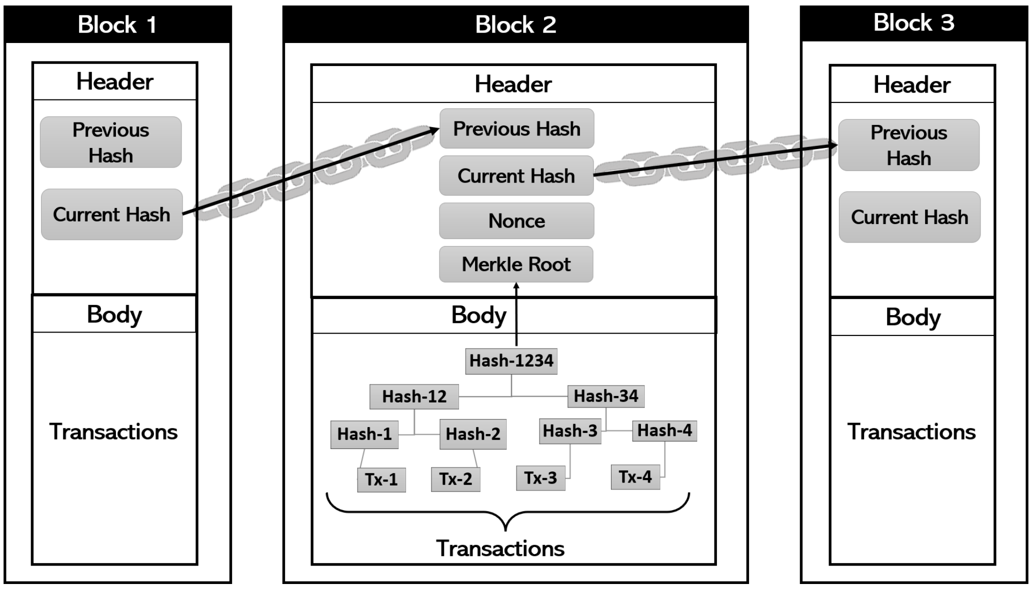


Figure 1. Blockchain General Architecture.

SCs are a crucial blockchain technological innovation that has the potential to change the way we conduct business. They make it possible to automate complex financial transactions without intermediaries such as banks or attorneys. SCs are self-executing programs recorded on a blockchain that run automatically when certain criteria are satisfied. They are secure and trustworthy since they are tamper-proof and transparent.

2.2. Application Domains

The widespread adoption of blockchain and smart contract technologies might significantly alter several markets. In the financial sector, for instance, blockchain technology can be utilized to build a safer and more reliable payment system. By eliminating the need for middlemen and drastically cutting down on transaction fees, SCs are revolutionizing the way business is conducted. To further expand people's and enterprises' access to financial services, blockchain technology can be utilized to build decentralized lending and investing platforms.

The real estate market is another sector that could profit from blockchain [28,29]. By eliminating middlemen and increasing accountability, blockchain technology has the capacity to revolutionize the property registration and transfer industry. If a sale is finalized or money is transferred, for instance, an SC can automatically transfer ownership to the buyer. For both parties involved, this can be a time- and cost-saving measure.

The healthcare sector is another that might greatly profit from blockchain implementation [30,31]. A patient's medical history can be kept on the distributed ledger technology known as blockchain. In order to better coordinate care and lower the likelihood of medical errors, SCs can be used to automate the sharing of medical records between healthcare providers. In addition, blockchain technology can help monitor drug authenticity and tampering as they travel through the supply chain.

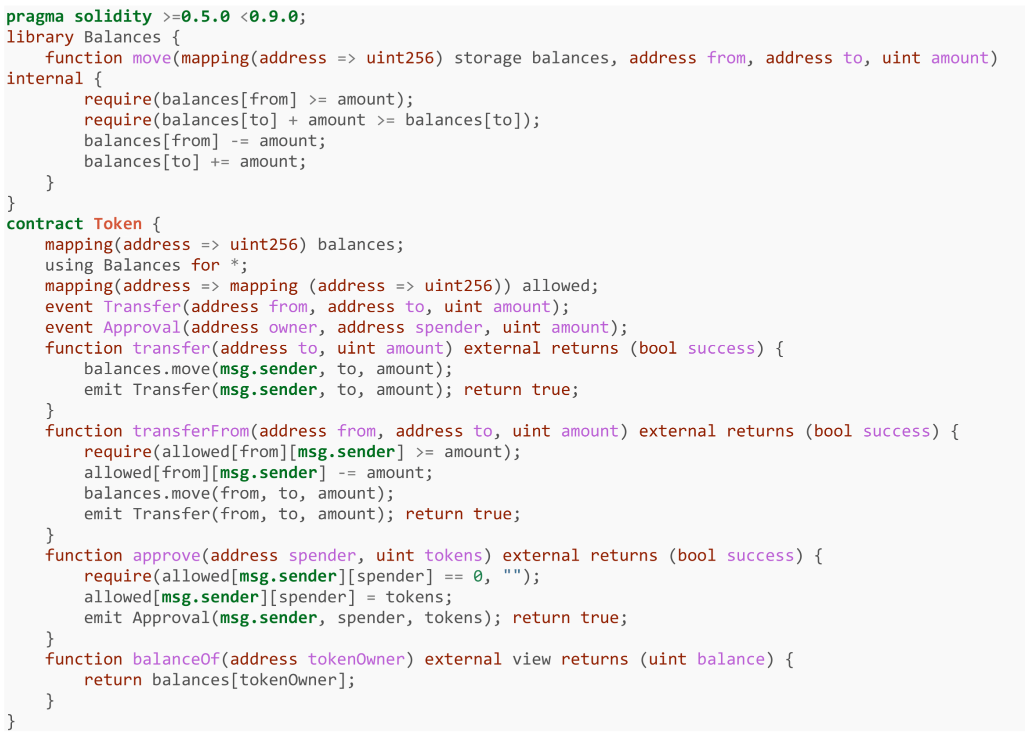
Blockchain technology also has potential in the supply chain sector [32,33]. Blockchain technology allows companies to keep tabs on product deliveries from start to finish, improving visibility while decreasing opportunities for fraud. Using SCs, the supply chain's financial transactions can be automated, eliminating the need for middlemen while increasing speed and accuracy.

Voting is another area where blockchain technology can be put to use [34,35]. With a blockchain-based voting system, we can eliminate the possibility of voter fraud and guarantee a fair and accurate tally of all votes cast. Vote tallying can be automated with the help of SCs, making the process quicker and more accurate.

A new revolution in the transportation sector called the Internet of Vehicles (IoV) has the ability to fix the problems with the established structure. The IoV's data security and privacy, however, present significant difficulties. Blockchain technology can solve the authentication problems of cars traveling from one trusted authority to another when combined with physical unclonable functions [36].

The widespread adoption of blockchain and smart contract technology might radically alter many sectors. Blockchain technology has the potential to enhance company operations, which in turn will benefit individuals and society at large, by making systems more secure, transparent, and efficient.

However, the security of SCs is a major concern because they are vulnerable to a variety of attacks, such as coding errors, malicious inputs, and blockchain network attacks. Securing SCs is critical to the viability of blockchain technology. Traditional techniques in SC security, such as code review and formal verification, are limited and may not always discover all sorts of vulnerabilities. As a result, there is increased interest in investigating the application of AI technology to improve SC security. By detecting anomalies and unusual behavior that may suggest a security breach, these techniques have the ability to provide a more thorough and proactive approach to security. We can improve trust and confidence in blockchain technology by increasing SC security and releasing its full potential for building a more decentralized and secure financial system. An SC developed in Solidity is presented in Figure 2.



```
pragma solidity >=0.5.0 <0.9.0;
library Balances {
    function move(mapping(address => uint256) storage balances, address from, address to, uint amount)
internal {
        require(balances[from] >= amount);
        require(balances[to] + amount >= balances[to]);
        balances[from] -= amount;
        balances[to] += amount;
    }
}
contract Token {
    mapping(address => uint256) balances;
    using Balances for *;
    mapping(address => mapping (address => uint256)) allowed;
    event Transfer(address from, address to, uint amount);
    event Approval(address owner, address spender, uint amount);
    function transfer(address to, uint amount) external returns (bool success) {
        balances.move(msg.sender, to, amount);
        emit Transfer(msg.sender, to, amount); return true;
    }
    function transferFrom(address from, address to, uint amount) external returns (bool success) {
        require(allowed[from][msg.sender] >= amount);
        allowed[from][msg.sender] -= amount;
        balances.move(from, to, amount);
        emit Transfer(from, to, amount); return true;
    }
    function approve(address spender, uint tokens) external returns (bool success) {
        require(allowed[msg.sender][spender] == 0, "");
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens); return true;
    }
    function balanceOf(address tokenOwner) external view returns (uint balance) {
        return balances[tokenOwner];
    }
}
```

Figure 2. An SC developed in Solidity.

Blockchain and SC solutions provide numerous advantages to consumers, businesses, and governments. Some are (Figure 3):

- Transparency: Decentralized blockchains are completely transparent. Transactions on the blockchain are transparent and verifiable. Nobody can also update network information. As a result, a user or company owner can create or use an SC without fear of a hacker altering it to steal money or data.
- Efficiency in the Economy: SCs automate numerous agreement-processing commercial activities. SCs do not require the services of attorneys, banks, or brokers. Both provide for significant cost reductions.
- Time-Saving Autonomy: Writing and monitoring a standard contract takes time. SCs are simpler and faster to implement: the programmer writes the contract code once and then utilizes it any time it is required (such as when trying to construct an NFT or for automatically filling out a bill and making trades).
- Building Trust: There are no humans among the SCs. This builds long-term trust amongst counteragents. If something goes wrong, the parties will look into it together.
- Safe Backup: Since businesses and governments risk losing important data, everyone copies it and backs it up. Even the most secure backup mechanisms cannot ensure data preservation. Hackers can either succeed or fail. Blockchain and SCs differ in that data are stored on several devices until the blockchain functions.
- Fraud Prevention: SCs prevent fraudulent access if the blockchain code is correct. Phishing can be prevented with time.
- Safety and Dependability: SCs are well known for their data security and market-leading encryption in IT. Blockchain and SC agreements are the most secure contracts available today.

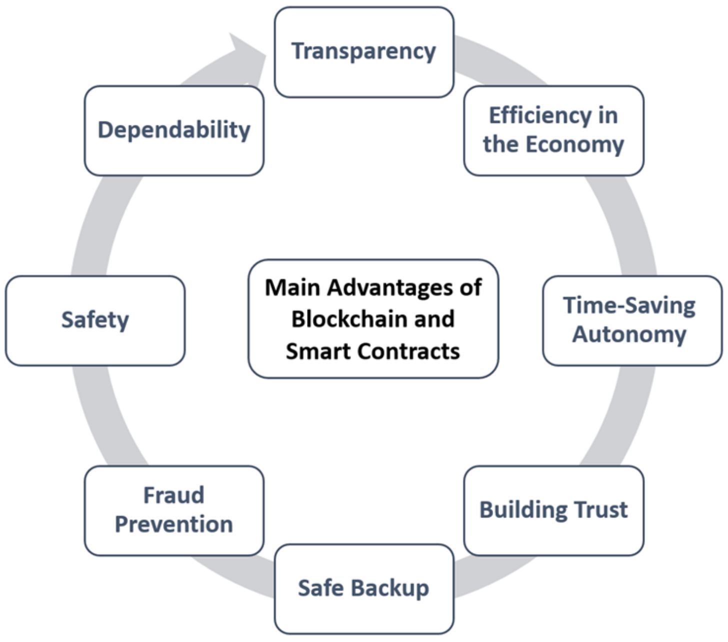


Figure 3. The main advantages of Blockchain and SCs.

2.3. Possible Attacks

There is a wide variety of threats that can target SCs and cause financial loss and reputational harm to blockchain networks. These attacks include (Table 1):

- Reentrancy attacks: An attacker can use this vulnerability to drain the contract's cash by repeatedly calling an SC function before the previous call has completed. An attacker can steal money from a contract by making a malicious contract that repeatedly calls the function of the target contract before the target contract has finished processing the previous call.

- Integer overflow and underflow attacks: These exploits make use of flaws in the way SCs handle integer values to subvert the contract's logic and steal money. By sending a huge negative number to a contract function that expects a positive number, the attacker can trigger an underflow and gain access to a significant amount of tokens.
- Denial-of-Service (DoS) attacks: The goal of these assaults is to prevent an SC from handling valid transactions by overwhelming its resources. This can be achieved by flooding the contract with a large number of little transactions or by sending transactions with input data that are too large and so exceed the contract's gas limit.
- Malicious input attacks: For these kinds of attacks, the attacker sends malicious data inputs to an SC in order to manipulate its behavior and, in the worst-case scenario, steal money. Input data sent by an attacker, for instance, could trigger a contract to transmit funds to an unauthorized address.
- Front-running attacks: To perform a front-running assault, one must take advantage of the brief window of opportunity between a transaction's submission and confirmation on the blockchain. An adversary can profit from this vulnerability by watching the blockchain for pending transactions and then submitting their own transaction with a greater gas price.
- Logic bombs: A logic bomb is malicious code that waits in an SC until a certain trigger condition is met, at which point the code is activated. For instance, an adversary can craft a contract that, at first glance, appears to work as intended, but actually contains malicious code that, after a certain date or time is reached, transfers funds to the adversary's account.
- Cross-chain attacks: These assaults take advantage of flaws in the way several blockchains communicate with one another. By taking advantage of differences in how several networks handle cross-chain transactions, an attacker can take cash from one network and move it to another.
- Time manipulation attacks: Time manipulation attacks take advantage of how SCs process information about the passage of time. An attacker may be able to cause a contract to execute too soon or wait forever if it depends on a timestamp or block number to trigger a certain action.
- Authorization flaws: A breach in authorization occurs when an SC does not adequately verify the identities of those who access the contract. An adversary could potentially use this flaw to conduct fraudulent transactions or gain access to the contract's cash.
- Gas limit attacks: In order to execute a contract, SCs must perform a certain amount of computational labor, which is measured in gas. An attacker can cause a transaction to fail by setting a low gas limit, resulting in the contract running out of gas before its execution is complete. The attacker may then be able to undo the transaction and steal money from the contract.

Table 1. Possible Types of Attacks.

Attack Type	**Description**	**Example**
Reentrancy Attacks	Allows an attacker to repeatedly call an SC function before the previous call completes	Drain a contract's funds by creating a malicious contract that calls the target contract's function multiple times
Integer Overflow and Underflow	Exploits vulnerabilities in the way SCs handle integer values	Underflow a contract's balance by sending a large negative number as input to a function expecting a positive number

Table 1. *Cont.*

Attack Type	Description	Example
Denial-of-Service (DoS)	Aims to overload an SC's resources, making it unable to process legitimate transactions	Send a large number of transactions to the contract in a short period of time
Malicious Input	Involves sending malicious data inputs to an SC, causing it to behave in unintended ways	Transfer funds to an unintended recipient by sending malicious input data to a contract
Front-Running	Involves exploiting the time delay between a transaction being submitted and confirmed on the blockchain	Profit from a transaction by submitting a higher gas price transaction ahead of the original transaction
Logic Bombs	A piece of malicious code that lies dormant in an SC until a specific trigger condition is met	Transfer funds to the attacker's account when a specific date or time is reached
Cross-Chain Attacks	Exploit vulnerabilities in the interaction between different blockchain networks	Steal funds from one network and transfer them to another by exploiting weaknesses in cross-chain transactions
Time Manipulation	Exploits the way SCs handle time-based events	Trigger an action prematurely or delay it indefinitely by manipulating the timestamp or block number
Authorization Flaws	Occurs when an SC fails to properly authenticate and authorize users who interact with it	Gain unauthorized access to a contract's funds or execute unauthorized transactions
Gas Limit Attacks	Exploit the way SCs handle gas, the unit of measurement for computational work	Revert a transaction and potentially steal funds by setting a low gas limit on a transaction

3. Background on Artificial Intelligence

3.1. Artificial Intelligence

Artificial intelligence (AI) is a vast area that includes the creation of intelligent computers capable of performing activities that normally require human intelligence [37,38]. Machine learning (ML) is a branch of AI that focuses on developing systems that can learn from data and make decisions without being explicitly programmed [39,40].

Developers use traditional programming techniques to manually write code to tackle a specific problem. The code is composed of a set of rules and instructions that the computer uses to generate output. This method necessitates a significant amount of human labor and is limited by the programmer's ability to predict all conceivable circumstances and edge cases. In contrast, with machine learning and artificial intelligence (AI), the computer is trained on a big dataset and learns to recognize patterns and make predictions or judgments based on that data. This method is more adaptable to new data and situations that were not explicitly programmed. The difference between Classical Programming and AI is illustrated in Figure 4.

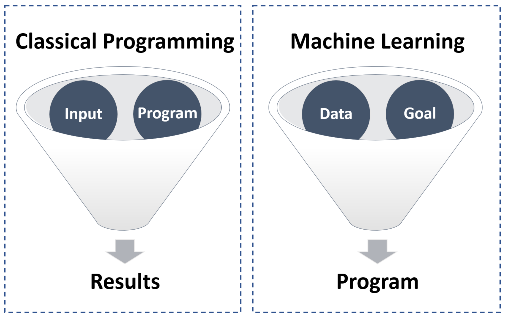


Figure 4. The difference between Classical Programming and Machine Learning.

The different phases of ML and AI include (Figure 5):

- Data collection: Collecting and preparing a large dataset that represents the problem domain.
- Data preprocessing: Cleaning and transforming the data to make it usable for ML models.
- Model selection and training: Choosing an appropriate ML model and training it on the dataset.
- Model evaluation: Evaluating the performance of the model on a separate dataset to measure its accuracy and effectiveness.
- Deployment: Implementing the model in a production system and integrating it with other systems as needed.
- Monitoring and maintenance: Continuously monitoring the model's performance and making updates and improvements as necessary.

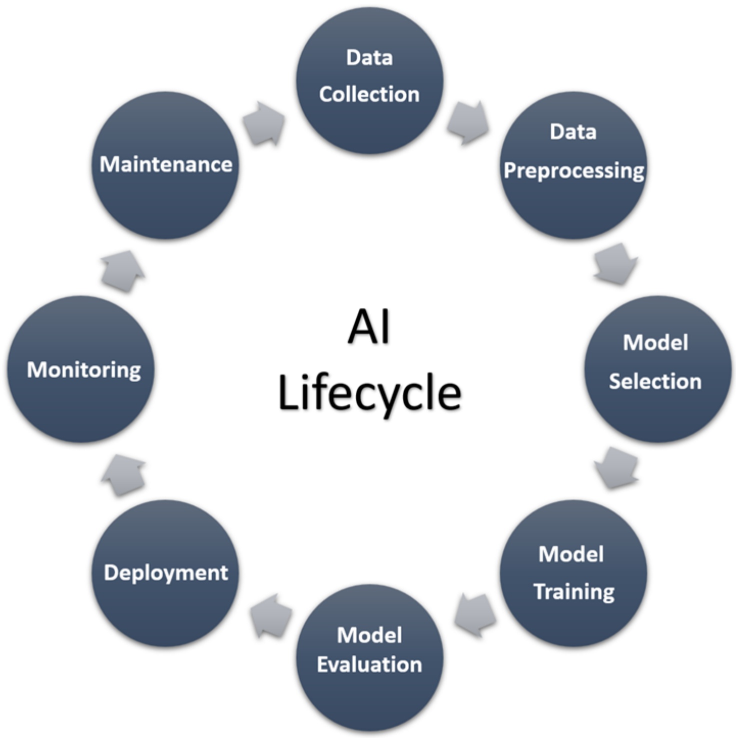


Figure 5. An illustration of AI lifecycle.

In sum, AI and ML are potent resources that can automate a wide range of jobs and generate highly accurate predictions and choices. They excel in areas where conventional programming methods would be too time-consuming or inefficient, such as those involving complexity.

3.2. Different Types of AI

There are four main types of AI [41] (Table 2):

- Supervised learning (SL) [42,43]: Supervised learning involves training an algorithm on a labeled dataset in which each input is accompanied by its corresponding label. The goal is to train the algorithm to correctly anticipate an output for inputs it has never seen before. Image categorization, voice recognition, and language translation are all applications of supervised learning.
- Unsupervised learning (USL) [44,45]: Unsupervised learning involves training an algorithm on a dataset without an associated label or output. The purpose of the algorithm is to autonomously identify such links or patterns in the data. Clustering, anomaly detection, and dimensionality reduction are all types of unsupervised learning.
- Semi-supervised learning (SSL) [46,47]: In semi-supervised learning, the algorithm is trained on a dataset with just some of the input data coupled with the right output or label. The goal is to train the algorithm to correctly predict an output for input data that it has never seen before, using both the labeled and unlabeled data to help it.
- Reinforcement learning (RL) [48,49]: In reinforcement learning, the algorithm learns to make decisions through trial and error. The algorithm receives feedback in the form of rewards or punishments for its actions and adjusts its behavior to maximize the reward. Examples of reinforcement learning include game playing, robotics, and autonomous vehicles.

Table 2. Different Types of AI.

Type of AI	**Description**	**Examples**
Supervised Learning	Trained on labeled data to predict correct output for new, unseen input data.	Image classification, speech recognition, language translation
Unsupervised Learning	Trained on unlabeled data to discover patterns or relationships in the data.	Clustering, anomaly detection, dimensionality reduction
Semi-Supervised Learning	Trained on partially labeled data to predict correct output for new, unseen input data while discovering patterns or relationships in the data.	Object recognition, speech recognition, sentiment analysis
Reinforcement Learning	Learns to make decisions through trial and error and adjusts behavior to maximize reward.	Game playing, robotics, autonomous vehicles

Each type of AI has its own strengths and weaknesses, and the choice of which type to use depends on the problem being solved and the available data. SL is useful when there is a well-defined output or label, USL is useful when there are no labeled data available, and RL is useful when the algorithm needs to learn through trial and error.

3.3. AI for Cybersecurity in General

The use of AI is increasingly vital in the field of cybersecurity [50]. By analyzing massive volumes of data and discovering patterns that suggest possible hazards, AI can be used to detect and prevent cyber-attacks [51,52]. Artificial intelligence algorithms can be taught to recognize common forms of assault and to spot novel ones that have certain traits. Patching security holes, keeping tabs on network traffic, and handling incidents are just some of the mundane responsibilities that may be automated with the help of AI [53,54]. Cybersecurity experts can then devote their time and energy to solving problems that call for their unique set of skills. Artificial intelligence (AI) has great potential in this area, but it cannot yet replace humans in this field. False positives and negatives can be avoided if artificial intelligence systems are trained, validated, and monitored effectively. Protecting AI systems from outside threats and ensuring their own safety is equally important. In general, AI has the ability to vastly improve cybersecurity by spotting and avoiding threats more quickly and correctly than before [55].

4. AI for Smart Contract Security

Significant monetary losses have resulted from several high-profile events involving hacked SCs in recent years. The application of AI technology, however, can improve smart contract security and reduce the likelihood of such instances occurring [56,57].

Using AI algorithms to examine the code and find flaws is one method of using AI to increase the safety of SCs. Patterns and abnormalities in the code that could suggest a security weakness can be trained into AI algorithms. AI algorithms can analyze enormous quantities of code to find common flaws, which can then be fixed in subsequent iterations of SCs [58].

A temporal message propagation network for extracting graph features was proposed by [59], who also investigated the application of graph neural networks and expert knowledge to discover vulnerabilities. Using a multi-layer bidirectional Transformer structure and using CodeBERT, VDDL was introduced by [60]. The multi-modal AI framework developed by [61] incorporated NLP, IR, and coding analysis methods. Using active SSL to combat the problem of insufficient labeled data and relying on bidirectional encoder representations from Transformers (BERT), Ref. [62] proposed an SC vulnerability detection system called ASSBert. By combining vulnerability identification and location into a single debugging process, Ref. [63] suggested a two-stage SC debugger dubbed ReVulDL, which employs a deep learning-based technique to detect and locate reentry vulnerabilities [64,65].

The use of natural language processing (NLP) techniques is yet another AI-based method for making SCs more secure. To prevent hackers from taking advantage of loopholes in SCs, natural language processing algorithms can examine the contracts' language for potential ambiguities and inconsistencies. Developers can improve the contract's safety by fixing these problems early on.

The authors of [66] developed a vulnerability detection model that included a hierarchical attention mechanism and made use of neural networks in AI, notably, BiLSTM (BiLSTM = bidirectional long short-term memory networks). To detect Ponzi schemes at the time of SC formation, Ref. [67] created a larger dataset and extracted various independent features from multiple views using a multi-view cascading ensemble model (MulCas). A heterogeneous graph transformation network for SC anomaly detection (SHGTNs) was proposed by [68] to identify instances of financial fraud on the Ethereum network. Using the bytecode of SCs as a new feature and GRU networks and attention mechanisms to obtain hidden information, Ref. [69] introduced SCSGuard, a framework that applied AI to identify fraudulent conduct in SCs.

Artificial intelligence can also be used to review SCs in real time, looking for signs of fraud or other irregularities. In an SC, for instance, AI algorithms can be taught to track the flow of money and flag any unusual activity. If developers are able to identify and report these transactions, they can take preventative measures.

The problem of local information loss in conventional CNN models was solved by [70] when they unveiled their new CNN architecture, CodeNet, for detecting flaws in SCs. To better uncover vulnerabilities in SCs, Ref. [71] used deep reinforcement learning in combination with multi-agent fuzz testing [72]. To determine whether SCs are vulnerable, Ref. [73] developed three distinct deep learning (DL) models: GRU, ANN, and LSTM. In order to discover flaws in SCs, Ref. [74] introduced a novel model called Link-DC, which uses deep and cross networks to build high-order nonlinear characteristics. By extracting features from both the high-level syntactic features and the low-level bytecode features of the SCs, the SmartMixModel vulnerability detection model presented by [75] improves the accuracy with which vulnerabilities in SCs can be identified.

In addition, AI can be utilized to build trustworthy and distributed SC administration infrastructure. Using AI to validate transactions and stop fraudulent conduct is one way to improve the safety of blockchain technology, the foundation of SCs. SCs can be made more hacker- and exploitation-resistant if decentralized systems are built utilizing AI.

An AI approach named GVD-net was suggested by [76] to identify flaws in Ethereum SCs. Ref. [77] introduced a static analysis tool for SCs using AI called Eth2Vec, which compared the target contract's code to a database of known vulnerable contract features learned automatically by neural networks. Ref. [78] developed a systematic and modular vulnerability detection framework based on DL, named DeeSCVHunter, for reentrancy and time-dependence vulnerabilities, while [79] used DL techniques to detect vulnerabilities in SCs by combining different representations of the code. To help find vulnerabilities in SCs, Hao et al. proposed SCscan, a scanning technique built on Support Vector Machines.

As a result, AI has the ability to greatly improve the security of SCs. Developers can limit the danger of hacking and exploitation by utilizing AI algorithms to examine code, natural language processing algorithms to detect linguistic ambiguities, and real-time monitoring of SCs. Furthermore, blockchain technology can be made more secure and transparent by developing decentralized systems for managing SCs. As SCs become more common, AI will play a growing role in guaranteeing their security and preventing fraudulent activities. A summary of the studies presented in this section is given in Table 3.

Table 3. Summary of Main Findings.

Ref.	Adopted Technique	Contribution
[59]	GNN, Expert Knowledge, Temporal Message Propagation Network	Proposed a technique for vulnerability detection in SCs using graph neural networks and expert knowledge, and introduced a temporal message propagation network to extract graph features
[60]	Multi-layer bidirectional Transformer structure, CodeBERT	Introduced VDDL, a vulnerability detection model that used a multi-layer bidirectional Transformer structure and incorporated CodeBERT
[61]	NLP, Image Processing, Code Analysis Techniques	Used a multi-modal AI framework for vulnerability detection in SCs
[62]	Active and SSL, BERT	Introduced ASSBert, an SC vulnerability detection framework that combines active SSL and employs BERT
[63]	DL-based Approach	Proposed ReVulDL, a two-stage SC debugger that uses a DL-based approach to detect and locate re-entry vulnerabilities
[66]	NN, BiLSTM, Hierarchical Attention Mechanism	Proposed a vulnerability detection tool that utilized neural networks and introduced a hierarchical attention mechanism
[67]	Multi-view Cascading Ensemble Model (MulCas)	Constructed a larger dataset and extracted numerous independent features from multiple perspectives for identifying Ponzi schemes when SC are created
[68]	Heterogeneous Graph Transformation Network	Proposed SHGTNs, a heterogeneous graph transformation network for detecting financial frauds on Ethereum platforms
[69]	ML, Bytecode, GRU Networks, Attention Mechanisms	Developed SCSGuard, a tool that used ML technology for detecting fraudulent behaviors in SCs by leveraging the bytecode of SCs as a novel feature
[70]	Convolutional Neural Network (CNN) Architecture	Introduced CodeNet, a new CNN architecture for detecting SC vulnerabilities that solved the problem of loss of local information in existing CNN models

Table 3. *Cont.*

Ref.	Adopted Technique	Contribution
[71]	Deep Reinforcement Learning, Multi-Agent Fuzz Testing	Developed improved techniques for detecting vulnerabilities in SCs using deep reinforcement learning and multi-agent fuzz testing
[73]	DL Models, LSTM, ANN, GRU	Trained three different DL models, GRU, ANN, LSTM and , and used them for predicting the existence of vulnerabilities in SCs
[74]	Deep and Cross Networks	Presented Link-DC, a new SC vulnerability detection model that used deep and cross networks for constructing high-order nonlinear features
[75]	High-level Syntactic Features, Low-level Bytecode Features	Introduced SmartMixModel, a vulnerability detection model that extracts features on two levels: low-level bytecode features and high-level syntactic features
[76]	AI Model	Proposed GVD-net, an AI model to detect security vulnerabilities in Ethereum SCs
[77]	AI-based Static Analysis Tool, Eth2Vec	Introduced Eth2Vec, an AI-based static analysis tool that utilized neural networks for automatically learning features of vulnerable contracts and detecting vulnerabilities in SCs
[79]	DL, Various Code Representations	Utilized DL techniques for detecting vulnerabilities in SCs by combining various code representations
[78]	DL, Modular and Systematic Vulnerability Detection Framework	Proposed DeeSCVHunter, a modular and systematic vulnerability detection framework based on DL, for reentrancy and time dependence vulnerabilities
[80]	SVM	Proposed SCscan, a scanning tool based on SVM for identifying potential security risks in SCs

5. Recommendations for Developers

Here are some concrete tips for developers about the use of artificial intelligence to improve smart contract security:

- Use AI for automated vulnerability detection: AI-based techniques can be used by developers to find flaws in SCs. These programs can examine code and detect potential security flaws. This can assist developers in finding and fixing vulnerabilities faster more effectively than manual testing.
- Use artificial intelligence to discover anomalies: AI may be used to monitor SCs and detect unusual behavior. A smart contract, for example, may suggest a security compromise if it suddenly begins performing a large number of transactions or accessing unexpected data. AI-based anomaly detection can aid in the rapid identification and response to these situations.
- AI may be used for predictive analytics to examine data from SCs and detect future security issues. For example, if a smart contract is utilized in a novel way, AI may analyze the data to predict whether this novel usage pattern is likely to result in security issues.
- AI can be utilized for behavior-based security by monitoring the behavior of SCs and detecting suspicious behavior. For example, if a smart contract begins to behave abnormally, AI can flag it for further examination.

Table 4 summarizes the advantages of AI-based security over classical techniques, which include:

- Efficiency: Since AI-based technologies can analyze code and data far more quickly than humans can, software engineers are able to locate and resolve issues much more quickly.
- Accuracy: AI has the ability to analyze massive amounts of data and recognize patterns that people would miss. This has the potential to result in improved vulnerability identification and more accurate predictive analytics.

- Scalability: AI-based tools can scan vast volumes of SCs at once, enabling developers to detect problems in a large number of contracts quickly. This is made possible through scalability.
- Adaptability: AI-powered technologies may learn from fresh data and adapt to new threats over time, making them more effective.

Table 4. Comparison of Classical and AI-based Security Techniques for SCs.

	Classical Techniques	**AI-Based Techniques**
Vulnerability Detection	Manual code review and testing	Automated code analysis and vulnerability detection
Anomaly Detection	Manual monitoring and analysis	Automated behavior-based anomaly detection
Predictive Analytics	Limited predictive capabilities	Advanced predictive analytics using machine learning
Behavior-based Security	Limited behavior-based monitoring	Advanced behavior-based monitoring using machine learning
Advantages	Established techniques, but slower and less accurate than AI-based techniques	Faster, more accurate, scalable, adaptable, and able to learn from new data and threats

In general, the use of AI-based techniques has a number of major advantages over the use of traditional techniques when it comes to improving the safety of SCs. To make their SCs more secure, developers should seriously consider employing AI-based technologies for vulnerability discovery, anomaly detection, predictive analytics, and behavior-based security.

6. Analysis, Findings, and Open Issues

AI approaches such as SL, SSL, and RL provide numerous advantages in SC vulnerability detection. SL is the most commonly used technique for effective pattern identification and feature extraction because of its ability to train on massive amounts of labeled data. This technique, however, has drawbacks, such as the requirement for a significant amount of labeled training data. SSL, albeit less widely utilized, offers the ability to alleviate this constraint by not requiring labeled training data in the pre-training phase. The potential of this technique to capture specific vulnerability features is its limitation. Due to the difficulty in collecting specific vulnerability features, USL approaches are rarely used in SC vulnerability identification. RL can be used to learn from the system's rewards or penalties, although more research is needed to determine its usefulness in SC security detection.

The use of AI in conjunction with fuzz testing, dynamic analysis, and static analysis approaches can also increase SC security detection. Static analysis uses information extracted from an SC's source code or bytecode for training AI models to find possible vulnerabilities. Dynamic analysis, on the other hand, records runtime data during contract execution in order to identify possible vulnerabilities and anomalous behaviors. Fuzz testing techniques generate random input data, and contract execution outcomes are evaluated to identify new vulnerabilities or abnormal behavior. However, there are still open issues, such as:

- Adversarial attacks: SCs are prone to adversarial attacks, in which attackers purposefully introduce malicious code or inputs to exploit contract weaknesses. Adversarial attacks provide a substantial barrier for AI-based SC security detection approaches because attackers can manipulate training data or circumvent detection by providing inputs targeted to elude detection. Future research should concentrate on creating more robust AI models capable of detecting adversarial attacks.
- Data privacy: AI-based SC vulnerability detection technologies necessitate access to massive volumes of data, raising privacy issues among users. SCs frequently contain sensitive information, such as financial transactions or personal data, which may be

exposed if data are not anonymized or protected adequately. Future research should concentrate on building privacy-preserving AI algorithms for detecting vulnerabilities in SCs while protecting user privacy.

- Scalability: AI-based SC security detection solutions will require scalability as the number of SCs on blockchain networks continues to expand. Due to the extensive time and computing power needed to train on huge datasets, scalability is a major issue for AI systems. The increasing volume of SC data necessitates the development of more effective and scalable AI solutions in the future.
- Interpretability: It can be difficult for users to comprehend the reasoning behind AI-based SC security detection models due to a lack of interpretability. Trust in the system is vital for the success of AI, and interpretability is the key to assuring the transparency and accountability of AI models. The next step in SC security research should be to create AI models that are easier to interpret for end users.
- Integration challenges: Integrating AI with formal methodologies presents substantial obstacles, but the benefits of doing so for SC security seem promising. Verifying the accuracy of an SC can be accomplished using either formal approaches, which use mathematical proofs and logical reasoning, or artificial intelligence (AI) methods, which utilize statistical models and AI algorithms to spot trends and outliers in the data. Developing formal models that can handle large-scale data and introducing AI techniques into the formal verification process are only two of the many technical hurdles that must be cleared in order to successfully merge these two approaches. More thorough and powerful SC security analysis tools cannot be created until these integration issues are resolved in future studies.

7. Conclusions and Future Work

Finally, there is still a gap in the current body of research regarding a comprehensive assessment of AI-based SC flaw detection despite the fact that artificial intelligence (AI) has made some strides in this area. By comparing and assessing existing AI-based SC fault detection algorithms, this work provides useful insights for scholars and practitioners. The paper also explores and contrasts the efficacy of various artificial intelligence approaches. While the potential of combining AI and formal approaches has been recognized, more research is required to address open issues and explore said potential.

The future of SC-related research will focus on the creation of AI-powered detection tools for SC-related security breaches that can handle ever-increasing amounts of data. These methods will be more productive and useful. SSL and RL, which have the potential to overcome the constraints of SL, should also be given more attention. Furthermore, there is a requirement for a comprehensive study into SC flaw detection using AI, which can serve as both a point of reference and a source of ideas for future research. To sum up, a more rigorous and comprehensive security research of SCs may be achieved by the integration of AI with formal techniques.

Funding: This research received no external funding.

Data Availability Statement: All data were presented in the main text.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* **2020**, *105*, 475–491. [CrossRef]
2. Derhab, A.; Guerroumi, M.; Belaoued, M.; Cheikhrouhou, O. BMC-SDN: Blockchain-based multicontroller architecture for secure software-defined networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 9984666. [CrossRef]
3. Dannen, C. *Introducing Ethereum and Solidity*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 1.
4. Wohrer, M.; Zdun, U. Smart contracts: Security patterns in the ethereum ecosystem and solidity. In Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Campobasso, Italy, 20 March 2018; IEEE: Piscataway, NJ, USA , 2018; pp. 2–8.

5. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
6. Baliga, A.; Solanki, N.; Verekar, S.; Pednekar, A.; Kamat, P.; Chatterjee, S. Performance characterization of hyperledger fabric. In Proceedings of the 2018 Crypto Valley conference on blockchain technology (CVCBT), Zug, Switzerland, 20–22 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 65–74.
7. Gorenflo, C.; Lee, S.; Golab, L.; Keshav, S. FastFabric: Scaling hyperledger fabric to 20000 transactions per second. *Int. J. Netw. Manag.* **2020**, *30*, e2099. [CrossRef]
8. Mohanty, D.; Mohanty, D. Corda architecture. In *R3 Corda for Architects and Developers: With Case Studies in Finance, Insurance, Healthcare, Travel, Telecom, and Agriculture*; Apress: New York, NY, USA, 2019; pp. 49–60.
9. Nadir, R.M. Comparative study of permissioned blockchain solutions for enterprises. In Proceedings of the 2019 International Conference on Innovative Computing (ICIC), Lahore, Pakistan, 1–2 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
10. Rouhani, S.; Deters, R. Security, performance, and applications of smart contracts: A systematic survey. *IEEE Access* **2019**, *7*, 50759–50779. [CrossRef]
11. Tsankov, P.; Dan, A.; Drachsler-Cohen, D.; Gervais, A.; Buenzli, F.; Vechev, M. Securify: Practical security analysis of smart contracts. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 67–82.
12. Sayeed, S.; Marco-Gisbert, H.; Caira, T. Smart contract: Attacks and protections. *IEEE Access* **2020**, *8*, 24416–24427. [CrossRef]
13. Atzei, N.; Bartoletti, M.; Cimoli, T. A survey of attacks on ethereum smart contracts (sok). In *Proceedings of the Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, 22–29 April 2017, Proceedings 6*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 164–186.
14. Krichen, M.; Maâlej, A.J.; Lahami, M. A model-based approach to combine conformance and load tests: An eHealth case study. *Int. J. Crit. Comput.-Based Syst.* **2018**, *8*, 282–310. [CrossRef]
15. Almakhour, M.; Sliman, L.; Samhat, A.E.; Mellouk, A. Verification of smart contracts: A survey. *Pervasive Mob. Comput.* **2020**, *67*, 101227. [CrossRef]
16. Bhargavan, K.; Delignat-Lavaud, A.; Fournet, C.; Gollamudi, A.; Gonthier, G.; Kobeissi, N.; Kulatova, N.; Rastogi, A.; Sibut-Pinote, T.; Swamy, N.; et al. Formal verification of smart contracts: Short paper. In Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, Vienna, Austria, 24 October 2016; pp. 91–96.
17. Krichen, M. Contributions to Model-Based Testing of Dynamic and Distributed Real-Time Systems. Ph.D. Thesis, École Nationale d'Ingénieurs de Sfax (Tunisie), Sfax, Tunisia, 2018.
18. Krichen, M.; Mihoub, A.; Alzahrani, M.Y.; Adoni, W.Y.H.; Nahhal, T. Are Formal Methods Applicable To Machine Learning And Artificial Intelligence? In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 9–11 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 48–53.
19. Yang, Z.; Lei, H.; Qian, W. A hybrid formal verification system in coq for ensuring the reliability and security of ethereum-based service smart contracts. *IEEE Access* **2020**, *8*, 21411–21436. [CrossRef]
20. Momeni, P.; Wang, Y.; Samavi, R. Machine learning model for smart contracts security analysis. In Proceedings of the 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, 26–28 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
21. Eshghie, M.; Artho, C.; Gurov, D. Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning. In Proceedings of the Evaluation and Assessment in Software Engineering, Trondheim, Norway, 21–23 June 2021; pp. 305–312.
22. Liao, J.W.; Tsai, T.T.; He, C.K.; Tien, C.W. Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing. In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 458–465.
23. Xing, C.; Chen, Z.; Chen, L.; Guo, X.; Zheng, Z.; Li, J. A new scheme of vulnerability analysis in smart contract with machine learning. *Wirel. Netw.* **2020**, 1–10. [CrossRef]
24. Namane, S.; Ahmim, M.; Kondoro, A.; Dhaou, I.B. Blockchain-Based Authentication Scheme for Collaborative Traffic Light Systems Using Fog Computing. *Electronics* **2023**, *12*, 431. [CrossRef]
25. Krichen, M.; Ammi, M.; Mihoub, A.; Almutiq, M. Blockchain for modern applications: A survey. *Sensors* **2022**, *22*, 5274. [CrossRef] [PubMed]
26. Namane, S.; Ben Dhaou, I. Blockchain-Based Access Control Techniques for IoT Applications. *Electronics* **2022**, *11*, 2225. [CrossRef]
27. Abbas, A.; Alroobaea, R.; Krichen, M.; Rubaiee, S.; Vimal, S.; Almansour, F.M. Blockchain-assisted secured data management framework for health information analysis based on Internet of Medical Things. *Pers. Ubiquitous Comput.* **2021**, 1–14. [CrossRef]
28. Latifi, S.; Zhang, Y.; Cheng, L.C. Blockchain-based real estate market: One method for applying blockchain technology in commercial real estate market. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 528–535.
29. Gupta, A.; Rathod, J.; Patel, D.; Bothra, J.; Shanbhag, S.; Bhalerao, T. Tokenization of real estate using blockchain technology. In *Proceedings of the Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, 19–22 October 2020, Proceedings 18*; Springer: Cham, Switzerland, 2020; pp. 77–90.

30. Agbo, C.C.; Mahmoud, Q.H.; Eklund, J.M. Blockchain technology in healthcare: A systematic review. *Healthcare* **2019**, *7*, 56. [CrossRef] [PubMed]
31. Hölbl, M.; Kompara, M.; Kamišalić, A.; Nemec Zlatolas, L. A systematic review of the use of blockchain in healthcare. *Symmetry* **2018**, *10*, 470. [CrossRef]
32. Dutta, P.; Choi, T.M.; Somani, S.; Butala, R. Blockchain technology in supply chain operations: Applications, challenges and research opportunities. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *142*, 102067. [CrossRef]
33. Chang, S.E.; Chen, Y. When blockchain meets supply chain: A systematic literature review on current development and potential applications. *IEEE Access* **2020**, *8*, 62478–62494. [CrossRef]
34. Taş, R.; Tanrıöver, Ö.Ö. A systematic review of challenges and opportunities of blockchain for E-voting. *Symmetry* **2020**, *12*, 1328. [CrossRef]
35. Kshetri, N.; Voas, J. Blockchain-enabled e-voting. *IEEE Softw.* **2018**, *35*, 95–99. [CrossRef]
36. Gupta, M.; Kumar, R.; Shekhar, S.; Sharma, B.; Patel, R.B.; Jain, S.; Dhaou, I.B.; Iwendi, C. Game Theory-Based Authentication Framework to Secure Internet of Vehicles with Blockchain. *Sensors* **2022**, *22*, 5119. [CrossRef]
37. Boulila, W.; Driss, M.; Alshanqiti, E.; Al-Sarem, M.; Saeed, F.; Krichen, M. Weight initialization techniques for deep learning algorithms in remote sensing: Recent trends and future perspectives. In *Advances on Smart and Soft Computing: Proceedings of ICACIn 2021*; Springer: Singapore, 2022; pp. 477–484.
38. Abdalzaher, M.S.; Salim, M.M.; Elsayed, H.A.; Fouda, M.M. Machine learning benchmarking for secured iot smart systems. In Proceedings of the 2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), Bali, Indonesia, 24–26 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 50–56.
39. Zidi, S.; Mihoub, A.; Qaisar, S.M.; Krichen, M.; Al-Haija, Q.A. Theft detection dataset for benchmarking and machine learning based classification in a smart grid environment. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 13–25. [CrossRef]
40. Hamdy, O.; Gaber, H.; Abdalzaher, M.S.; Elhadidy, M. Identifying exposure of urban area to certain seismic hazard using machine learning and GIS: A case study of greater Cairo. *Sustainability* **2022**, *14*, 10722. [CrossRef]
41. Zhang, C.; Lu, Y. Study on artificial intelligence: The state of the art and future prospects. *J. Ind. Inf. Integr.* **2021**, *23*, 100224. [CrossRef]
42. Cunningham, P.; Cord, M.; Delany, S.J. Supervised learning. In *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 21–49.
43. Hastie, T.; Tibshirani, R.; Friedman, J.; Hastie, T.; Tibshirani, R.; Friedman, J. Overview of supervised learning. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, NY, USA, 2009; pp. 9–41.
44. Hastie, T.; Tibshirani, R.; Friedman, J.; Hastie, T.; Tibshirani, R.; Friedman, J. Unsupervised learning. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, NY, USA, 2009; pp. 485–585.
45. Ghahramani, Z. Unsupervised learning. In *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, 2–14 February 2003, Tübingen, Germany, 4–16 August 2003, Revised Lectures*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 72–112.
46. Zhou, Z.H.; Zhou, Z.H. Semi-supervised learning. In *Machine Learning*; Springer: Singapore, 2021; pp. 315–341.
47. Van Engelen, J.E.; Hoos, H.H. A survey on semi-supervised learning. *Mach. Learn.* **2020**, *109*, 373–440. [CrossRef]
48. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement learning for combinatorial optimization: A survey. *Comput. Oper. Res.* **2021**, *134*, 105400. [CrossRef]
49. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
50. Dbouk, T.; Mourad, A.; Otrok, H.; Tout, H.; Talhi, C. A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1665–1680. [CrossRef]
51. Sarker, I.H.; Furhad, M.H.; Nowrozy, R. Ai-driven cybersecurity: An overview, security intelligence modeling and research directions. *SN Comput. Sci.* **2021**, *2*, 173. [CrossRef]
52. Dash, B.; Ansari, M.F.; Sharma, P.; Ali, A. Threats and Opportunities with AI-based Cyber Security Intrusion Detection: A Review. *Int. J. Softw. Eng. Appl. (IJSEA)* **2022**, *13*. [CrossRef]
53. Jaber, A.; Fritsch, L. Towards AI-powered Cybersecurity Attack Modeling with Simulation Tools: Review of Attack Simulators. In *Proceedings of the Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 17th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2022)*; Springer: Cham, Switzerland, 2022; pp. 249–257.
54. Ansari, M.F.; Dash, B.; Sharma, P.; Yathiraju, N. The Impact and Limitations of Artificial Intelligence in Cybersecurity: A Literature Review. *Int. J. Adv. Res. Comput. Commun. Eng.* **2022**. [CrossRef]
55. Srinivasan, S.; Ravi, V.; Sowmya, V.; Krichen, M.; Noureddine, D.B.; Anivilla, S.; Soman, K. Deep convolutional neural network based image spam classification. In Proceedings of the 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 4–5 March 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 112–117.
56. Demertzis, K.; Iliadis, L.; Tziritas, N.; Kikiras, P. Anomaly detection via blockchained deep learning smart contracts in industry 4.0. *Neural Comput. Appl.* **2020**, *32*, 17361–17378. [CrossRef]
57. Yunis, M.M.; El-Khalil, R.; Ghanem, M. Towards a Conceptual Framework on the Importance of Privacy and Security Concerns in Audit Data Analytics. In Proceedings of the International Conference on Industrial Engineering and Operations Management, Sao Paulo, Brazil, 5–8 April 2021.

58. Kumar, N.; Singh, A.; Handa, A.; Shukla, S.K. Detecting malicious accounts on the Ethereum blockchain with supervised learning. In *Proceedings of the Cyber Security Cryptography and Machine Learning: Fourth International Symposium, CSCML 2020, Be'er Sheva, Israel, 2–3 July 2020, Proceedings 4*; Springer: Cham, Switzerland, 2020; pp. 94–109.
59. Liu, Z.; Qian, P.; Wang, X.; Zhuang, Y.; Qiu, L.; Wang, X. Combining graph neural networks with expert knowledge for smart contract vulnerability detection. *IEEE Trans. Knowl. Data Eng.* **2021** . [CrossRef]
60. Jiang, F.; Cao, Y.; Xiao, J.; Yi, H.; Lei, G.; Liu, M.; Deng, S.; Wang, H. VDDL: A Deep Learning-Based Vulnerability Detection Model for Smart Contracts. In *Proceedings of the International Conference on Machine Learning for Cyber Security*; Springer: Cham, Switzerland, 2023; pp. 72–86.
61. Jie, W.; Chen, Q.; Wang, J.; Koe, A.S.V.; Li, J.; Huang, P.; Wu, Y.; Wang, Y. A novel extended multimodal AI framework towards vulnerability detection in smart contracts. *Inf. Sci.* **2023**, *636*, 118907. [CrossRef]
62. Sun, X.; Tu, L.; Zhang, J.; Cai, J.; Li, B.; Wang, Y. ASSBert: Active and semi-supervised bert for smart contract vulnerability detection. *J. Inf. Secur. Appl.* **2023**, *73*, 103423. [CrossRef]
63. Zhang, Z.; Lei, Y.; Yan, M.; Yu, Y.; Chen, J.; Wang, S.; Mao, X. Reentrancy Vulnerability Detection and Localization: A Deep Learning Based Two-phase Approach. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, Rochester, MI, USA, 10–14 October 2022; pp. 1–13.
64. Abdalzaher, M.S.; Soliman, M.S.; El-Hady, S.M.; Benslimane, A.; Elwekeil, M. A deep learning model for earthquake parameters observation in IoT system-based earthquake early warning. *IEEE Internet Things J.* **2021**, *9*, 8412–8424. [CrossRef]
65. Mihoub, A. A deep learning-based framework for human activity recognition in smart homes. *Mob. Inf. Syst.* **2021**, *2021*, 6961343. [CrossRef]
66. Xu, G.; Liu, L.; Zhou, Z. Reentrancy Vulnerability Detection of Smart Contract Based on Bidirectional Sequential Neural Network with Hierarchical Attention Mechanism. In Proceedings of the 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), Huaihua, China, 15–17 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 56–59.
67. Zheng, Z.; Chen, W.; Zhong, Z.; Chen, Z.; Lu, Y. Securing the Ethereum from Smart Ponzi Schemes: Identification Using Static Features. *ACM Trans. Softw. Eng. Methodol.* **2022**. [CrossRef]
68. Liu, L.; Tsai, W.T.; Bhuiyan, M.Z.A.; Peng, H.; Liu, M. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Gener. Comput. Syst.* **2022**, *128*, 158–166. [CrossRef]
69. Hu, H.; Bai, Q.; Xu, Y. Scsguard: Deep scam detection for ethereum smart contracts. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Virtual, 2–5 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
70. Hwang, S.J.; Choi, S.H.; Shin, J.; Choi, Y.H. CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. *IEEE Access* **2022**, *10*, 32595–32607. [CrossRef]
71. Andrijasa, M.F.; Ismail, S.A.; Ahmad, N. Towards Automatic Exploit Generation for Identifying Re-Entrancy Attacks on Cross-Contract. In Proceedings of the 2022 IEEE Symposium on Future Telecommunication Technologies (SOFTT), Johor Baharu, Malaysia, 14–16 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 15–20.
72. Kang, D. Bridging Fuzz Testing and Metamorphic Testing for Classification of Machine Learning. In Proceedings of the 2022 IEEE International Conference on Consumer Electronics (ICCE), Taipei, Taiwan, 6–8 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–2.
73. Gupta, R.; Patel, M.M.; Shukla, A.; Tanwar, S. Deep learning-based malicious smart contract detection scheme for internet of things environment. *Comput. Electr. Eng.* **2022**, *97*, 107583. [CrossRef]
74. Li, N.; Liu, Y.; Li, L.; Wang, Y. Smart Contract Vulnerability Detection Based on Deep and Cross Network. In Proceedings of the 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA), Changchun, China, 20–22 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 533–536.
75. Shakya, S.; Mukherjee, A.; Halder, R.; Maiti, A.; Chaturvedi, A. SmartMixModel: Machine Learning-based Vulnerability Detection of Solidity Smart Contracts. In Proceedings of the 2022 IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 22–25 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 37–44.
76. Wang, Z.; Zheng, Q.; Sun, Y. GVD-net: Graph embedding-based Machine Learning Model for Smart Contract Vulnerability Detection. In Proceedings of the 2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMIT), Xi'an, China, 23–25 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 99–103.
77. Ashizawa, N.; Yanai, N.; Cruz, J.P.; Okamura, S. Eth2Vec: Learning contract-wide code representations for vulnerability detection on ethereum smart contracts. In Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure, Virtual Event, Hong Kong, 7 June 2021; pp. 47–59.
78. Yu, X.; Zhao, H.; Hou, B.; Ying, Z.; Wu, B. Deescvhunter: A deep learning-based framework for smart contract vulnerability detection. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.

79. Wang, B.; Chu, H.; Zhang, P.; Dong, H. Smart Contract Vulnerability Detection Using Code Representation Fusion. In Proceedings of the 2021 28th Asia-Pacific Software Engineering Conference (APSEC), Taipei, Taiwan, 6–9 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 564–565.
80. Hao, X.; Ren, W.; Zheng, W.; Zhu, T. SCScan: A SVM-Based Scanning System for Vulnerabilities in Blockchain Smart Contracts. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December–1 January 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1598–1605.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.