Turkish Journal of Computer and Mathematics Education *Vol.11 No.03 (2020), 2010-2021*
DOI: https://doi.org/10.17762/turcomat.v11i3.13598
Research Article

Smart Contract Technology for Blockchain-based Systems: Design and Implementation Challenges.

Saumitra Chattopadhyay,

Department of Comp. Sc. & Info. Tech., Graphic Era Hill University, Dehradun, Uttarakhand, India 248002,

Abstract: The technology of smart contracts, which is an essential component of blockchain-based systems, has the potential to improve the efficiency, transparency, and safety of these systems. However, there are a number of challenges that need to be conquered in order to develop and implement smart contracts, particularly in regard to security, adaptability, interoperability, and meeting the standards imposed by legal and regulatory authorities. This paper examines ideas for designing and implementing smart contracts, as well as providing an outline of the challenges that have been encountered. In this article, we take a look back at 25 essential works that were published in the field between 2010 and 2019, and we discuss the impact that these works have had. According to the findings of our research, continual collaboration between the necessary stakeholders is required in order to fully realize the benefits given by smart contract technology while also limiting the risks associated with it.

Keywords: Security, adaptability, interoperability, best-practices, collaboration, efficiency, transparency, and risk management.

I. Introduction

Smart contracts, are those that may be designed to carry out the provisions of a contract between two parties when specific conditions are satisfied. These contracts are often created as computer code and are carried out using a blockchain-based platform, making them transparent and impermeable [1]. From finance to real estate to supply chain management, smart contract technology has the potential to disrupt several industries.Unfortunately, creating and executing smart contracts has several difficulties. Making sure the contract code is safe and free of flaws is one of the main problems. Once implemented, smart contracts are immutable, therefore any bugs or mistakes in the code cannot be fixed. Because of this, it is crucial to carefully test and audit smart contract code to make sure it is secure [2].Designing smart contracts that are adaptable enough to manage changing business regulations or conditions is another difficulty. Any modifications to the circumstances that a smart contract is supposed to perform must be made in a new contract, which must then be created and deployed. This can be costly and time-consuming, especially if the contract is intricate [3].Making sure smart contracts can communicate with other systems and contracts is a related challenge. Writing contracts that may be performed on several blockchain platforms can be challenging due to the potential existence of different contract languages and syntaxes on different blockchain platforms. Contracts may also need to communicate with outside systems or data sources, which might be challenging to integrate.The final need is that smart contracts be created to adhere to all applicable legal and regulatory frameworks. This can be difficult because there is sometimes uncertainty over the legal standing of smart contracts in various jurisdictions, thus developers must carefully evaluate any potential legal ramifications of the contracts they create.In conclusion, smart contract technology has the potential to revolutionize a variety of industries. However, designing and putting into practice flexible, secure contracts that can work with other systems and adhere to legal and regulatory frameworks presents a number of difficulties that must be carefully addressed [4].By enabling safe, open, and quick transactions, blockchain technology has the potential to completely transform many different industries. Smart contracts are one of the major advancements in blockchain technology. Self-executing contracts, or smart contracts, are those that may be designed to carry out the provisions of a contract between two parties when specific conditions are satisfied. Smart contracts are often created as computer code and run on a blockchain-based platform, making them transparent and impermeable.Although smart contract technology has many advantages, creating and using smart contracts has its share of difficulties. These difficulties include securing the smart contract code, creating adaptable contracts that can consider shifting business regulations and

conditions, assuring compatibility with other systems and contracts, and creating contracts that adhere to the relevant legal and regulatory frameworks [5].By doing this, we seek to encourage the use of smart contract technology and give stakeholders the opportunity to take full advantage of its potential advantages.By enabling safe, open, and quick transactions, blockchain technology has the potential to completely transform many different industries. Smart contracts are one of the major advancements in blockchain technology. Self-executing contracts, or smart contracts, are those that may be designed to carry out the provisions of a contract between two parties when specific conditions are satisfied. Smart contracts are often created as computer code and run on a blockchain-based platform, making them transparent and impermeable [6].Compared to conventional contract management systems, smart contract technology has several advantages. Smart contracts eliminate the need for middlemen and lower transaction costs by automatically executing when certain pre-specified conditions are met. Because the code and execution history of smart contracts are maintained on the blockchain, easy auditing and accountability are also made possible. Moreover, self-enforcing smart contracts can be created to make sure that all parties abide by the agreement's provisions.Unfortunately, creating and executing smart contracts has severaldifficulties[7]. The most difficult task is making sure the smart contract code is secure. Once implemented, smart contracts are immutable, therefore any bugs or mistakes in the code cannot be fixed. Because of this, it is crucial to carefully test and audit smart contract code to make sure it is secure. An attacker may be able to steal money or run harmful code as a result of a single vulnerability in a smart contract.Designing smart contracts that are adaptable enough to manage changing business regulations or conditions is another difficulty. Any modifications to the circumstances that a smart contract is supposed to perform must be made in a new contract, which must then be created and deployed. This can be costly and time-consuming, especially if the contract is intricate [8].Making sure smart contracts can communicate with other systems and contracts is a related challenge. Writing contracts that may be performed on several blockchain platforms can be challenging due to the potential existence of different contract languages and syntaxes on different blockchain platforms. Contracts may also need to communicate with outside systems or data sources, which might be challenging to integrate.The final need is that smart contracts be created to adhere to all applicable legal and regulatory frameworks. This can be difficult because there is sometimes uncertainty over the legal standing of smart contracts in various jurisdictions, thus developers must carefully evaluate any potential legal ramifications of the contracts they create. Smart contract technology has many advantages over conventional contract management systems, but creating flexible, secure contracts that can integrate with other systems and adhere to legal and regulatory requirements presents a number of difficulties that must be carefully resolved [9]. This essay seeks to give a general overview of these issues and suggest effective solutions. By doing this, we seek to encourage the use of smart contract technology.A significant advancement in blockchain technology that has the potential to transform many sectors is smart contract technology. Self-executing contracts, or smart contracts, are those that may be designed to carry out the provisions of a contract between two parties when specific conditions are satisfied. Smart contracts are often created as computer code and run on a blockchain-based platform, making them transparent and impermeable [10].Compared to conventional contract management systems, smart contract technology has a number of advantages. Eliminating intermediaries in transactions is one of the main advantages since it can lower transaction costs and boost efficiency. Moreover, smart contracts may be carried out automatically, eliminating the need for human participation and lowering the possibility of mistakes or legal challenges[11]. Because the code and execution history of smart contracts are maintained on the blockchain, easy auditing and accountability are also made possible.Moreover, self-enforcing smart contracts can be created to make sure that all parties abide by the agreement's provisions. This may lessen the need for expensive and drawn-out dispute resolution procedures. In order to reduce the need for manual intervention and boost productivity, smart contracts can also be set up to execute automatically when specific criteria are satisfied, such as the delivery of goods or the receipt of cash.Several industries, including finance, supply chain management, and real estate, can benefit from the adoption of smart contracts. Smart contracts can be used in the financial sector to automate the execution of financial transactions like loans and derivatives, thereby obviating the need for middlemen and improving efficiency. Smart contracts can be utilized in supply chain management to automate the tracking and verification of items, lowering the risk of fraud and raising transparency. Smart contracts can be used in real estate to automate the completion of property deals, obviating the need for middlemen and improving

efficiency.Unfortunately, creating and executing smart contracts has a number of difficulties. The most difficult task is making sure the smart contract code is secure. Once implemented, smart contracts are immutable, therefore any bugs or mistakes in the code cannot be fixed. Because of this, it is crucial to carefully test and audit smart contract code to make sure it is secure[12]. An attacker may be able to steal money or run harmful code as a result of a single vulnerability in a smart contract.Designing smart contracts that are adaptable enough to manage changing business regulations or conditions is another difficulty. Any modifications to the circumstances that a smart contract is supposed to perform must be made in a new contract, which must then be created and deployed. This can be costly and time-consuming, especially if the contract is intricate.Despite these difficulties, smart contract technology has a lot of potential advantages and is spreading throughout many different industries. The advantages of smart contract technology could become even more substantial as blockchain technology develops and is utilized by more people [13].

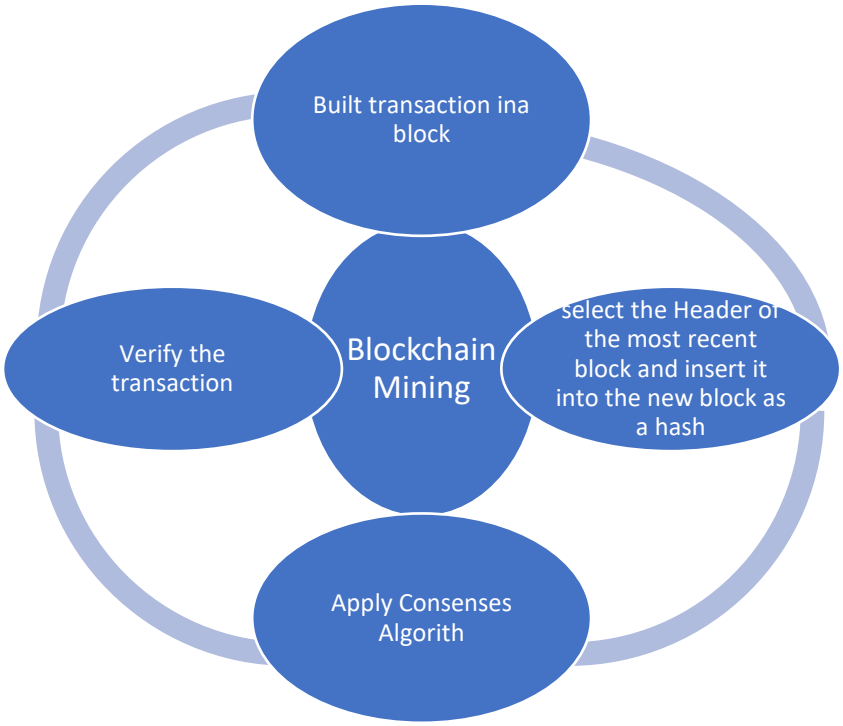

Figure 1. Blockchain Mining characteristic

A. Overview of the challenges in designing and implementing smart contracts

To ensure the security, adaptability, and interoperability of the contracts, a number of difficulties must be carefully addressed during the design and implementation of smart contracts. Among the principal difficulties are:

i. **Security:** Ensuring the security of smart contracts is one of the main difficulties in developing and putting them into use. Once implemented, smart contracts are immutable, therefore any bugs or mistakes in the code cannot be fixed. Because of this, it is crucial to carefully test and audit smart contract code to make sure it is secure. An attacker may be able to steal money or run harmful code as a result of a single vulnerability in a smart contract.

ii. **Flexibility:** As smart contracts are normally created to carry out a certain set of conditions, any modifications to those criteria necessitate the creation and deployment of a new contract. This can be costly and time-consuming, especially if the contract is intricate. It is consequently a huge problem to design smart contracts that are adaptable enough to accommodate changing business regulations or conditions.

iii. **Interoperability:** Writing contracts that can be performed on numerous blockchain platforms can be challenging since different blockchain platforms may have distinct contract languages and syntax. Contracts may also need to communicate with outside systems or data sources, which might be challenging to integrate. So, another key problem is ensuring that smart contracts can interact with other systems and contracts.

iv. **Legal and regulatory frameworks :**Smart contracts need to be created in accordance with all applicable legal and regulatory frameworks. This can be difficult because there is sometimes uncertainty over the legal standing of smart contracts in various jurisdictions, thus developers must carefully evaluate any potential legal ramifications of the contracts they create.

v. **Complexity:** Smart contracts can be complicated, particularly when several parties and intricate business regulations are involved. Complex smart contract design and implementation take a lot of knowledge and resources, which might be difficult for businesses that are new to the technology.While being programmed to run automatically, smart contracts are still developed by people and may include problems or errors. Therefore, it is crucial to make sure that smart contract code is adequately tested and audited to reduce the chance of errors.A substantial problem is in creating secure, adaptable, and interoperable smart contracts that adhere to legal and regulatory frameworks. Yet, corporations may realize the potential advantages of smart contract technology and transform numerous industries by thoughtfully addressing these problems.

II. Security Challenges in Smart Contract Design and Implementation

One of the biggest obstacles to designing and implementing smart contracts is security. A smart contract becomes immutable once it is published on a blockchain, which means that any mistakes or security holes in the code cannot be fixed. This necessitates that security be given top priority while designing and implementing smart contracts. In this section, we'll talk about some of the major security issues that arise during the development and use of smart contracts.

i. **Code Vulnerabilities**:Smart contracts are susceptible to programming flaws including buffer overflows, integer overflows, and input validation mistakes because they are written in code. Attackers may use these flaws to run malicious code, take money, or otherwise interfere with the functionality of the contract. To find and address vulnerabilities, it is crucial to fully test smart contract code and carry out frequent security audits.

ii. **Malicious Intent:**Smart contracts are susceptible to bad intent since they are performed automatically and without human involvement. Attackers may use flaws in smart contracts to execute code that wasn't intended to be executed, steal money, or otherwise interfere with the execution of the contract. Smart contract designers should use security measures like access controls, data encryption, and logging to reduce this risk.

iii. **Oracle manipulation:** Smart contracts could have to communicate with outside systems or data sources, which are susceptible to error. Attackers can modify data sources to provide the smart contract misleading information, which would cause it to run malicious code or take unexpected actions. To prevent oracle manipulation, smart contract designers should include security safeguards such data validation and verification.

iv. **Vulnerabilities of the blockchain:** A blockchain, which is used to execute smart contracts, is susceptible to attacks like 51% attacks, denial-of-service attacks, and consensus failures. The security of smart contracts operating on the blockchain may be jeopardized by these attacks. Designers of smart contracts should use a blockchain platform with a strong consensus mechanism and frequently check the blockchain for security risks to reduce this risk.

v. **Upgradeability:** Because smart contracts are immutable, it is difficult to fix any bugs or security holes in the code. Several smart contract developers have introduced upgradable smart contracts, which enable the contract's code to be modified without jeopardizing its security, as a solution to this problem.

The possibility of illegal code alterations is one of the additional security threats that upgradeable smart contracts potentially bring about.

Security poses a serious obstacle to the creation and use of smart contracts. The security issues associated with smart contracts, including as code vulnerabilities, bad intent, oracle manipulation, blockchain vulnerabilities, and upgradability, must be carefully considered and addressed by smart contract designers. Organizations can reduce the risk of security breaches and guarantee the secure operation of their smart contracts by installing strong security controls and performing frequent security audits.

A. Explanation of the importance of security in smart contract design

It is impossible to overestimate the significance of security in smart contract design. When deployed on a blockchain, smart contracts become immutable, which means that any bugs or vulnerabilities in the code cannot be fixed. Smart contracts are computer programs that run automatically and without human involvement. Because of this, it is essential to develop and implement smart contracts with a strong security focus.A security lapse in a smart contract may have serious repercussions. Bad actors may take advantage of weaknesses in smart contracts to run code that was not intended by the contract's designers, steal money, or otherwise interfere with the contract's functionality. Security flaws in smart contracts have occasionally resulted in substantial financial losses or even the demise of entire blockchain initiatives.The security issues related to smart contracts must be properly considered and addressed by smart contract designers in order to mitigate these risks. This entails carrying out exhaustive security audits, putting in place strong security measures, and following best practices for developing smart contracts.Code flaws are among the most important components of smart contract security. Programming issues like buffer overflows, integer overflows, and input validation flaws can affect smart contracts because they are written in code. Attackers may use these flaws to run malicious code, steal money, or cause havoc with the contract's operations. The code for smart contracts must be carefully tested, and frequent security audits must be performed to find and address flaws.Oracle manipulation is yet another crucial facet of smart contract security. It's possible that smart contracts will have to communicate with untrusted external systems or data sources. Attackers can force misleading data into the smart contract, causing it to run malicious code or take unexpected actions, by manipulating the data sources. For the purpose of preventing oracle manipulation, smart contract designers must include security controls like data validation and verification. It is impossible to exaggerate the value of security in the design of smart contracts. Designers of smart contracts must carefully evaluate and solve the security issues related to smart contracts, including code vulnerabilities, bad intent, oracle manipulation, blockchain vulnerabilities, and upgradability. Organizations can reduce the risk of security breaches and guarantee the safe and secure operation of their smart contracts by installing strong security controls and carrying out frequent security audits.

B. Discussion of common security vulnerabilities in smart contracts (e.g., reentrancy attacks, integer overflows

Because they are written in code, smart contracts are susceptible to a number of security flaws. Attackers may use these flaws to run malicious code, take money, or otherwise interfere with the functionality of the contract. We'll talk about some of the most prevalent smart contract security flaws in this part.

Attacks on reentrancy: A reentrancy vulnerability allows a smart contract to call another contract before the previous one has done executing, which is how reentrancy attacks work. As a result, an attacker may be able to repeatedly call a contract function to syphon money from it.

Integer Overflow & Underflow: When the outcome of an arithmetic operation exceeds the maximum or minimum value that the data type may represent, an integer overflow or underflow vulnerability exists. This may result in the smart contract behaving unexpectedly, and attackers may utilize this to run malicious code.

Access Control Problems: When a smart contract fails to limit access to specific functionalities or data to authorized parties, access control vulnerabilities happen. This could expose sensitive information or provide an attacker access to privileged software, jeopardizing the contract's security.

Time Manipulation: A smart contract becomes vulnerable to time manipulation when it uses the current time to make decisions or carry out operations. Attackers can change the functionality of the contract or alter the system time so that malicious code will run.

Malicious Libraries & Dependencies: Smart contracts frequently rely on external libraries and dependencies, which may have security flaws or be deliberately harmful. Attackers can take advantage of these flaws to run malicious code or alter the behaviour of the contract.

Gas Limitation Vulnerabilities: Gas limitation vulnerabilities happen when a smart contract under-estimates the quantity of gas required to carry out a function. Due to this, the contract may not perform as expected or may be exposed to denial-of-service attacks.Smart contract designers must include security controls like access restrictions, data encryption, and logging in order to mitigate these risks. To find and address vulnerabilities, frequent security audits and testing are also required.

An number of security flaws, including as reentrancy attacks, integer overflow and underflow, access control problems, time manipulation, malicious libraries and dependencies, and gas limits, make smart contracts susceptible. To ensure the safe execution of their contracts, smart contract designers must carefully analyze and solve these risks. Organizations can reduce the risk of security breaches and guarantee the secure operation of their smart contracts by installing strong security controls and performing frequent security audits.

C. Overview of methods for testing and auditing smart contract code

Blockchain-based systems depend on smart contracts, and any flaws in its design might have disastrous results, including money lost, business interruptions, and reputational harm. As a result, before deployment, it is essential to properly test and audit the smart contract code to find and fix any flaws. We will give an overview of the techniques for testing and auditing smart contract code in this part.

i. **Manual Code Review:** During a manual code review, a group of subject-matter experts examine the code line by line to find any potential security flaws. This method requires a lot of time and resources, but it has the potential to find complicated vulnerabilities that automated tools could overlook.
ii. **Static Analysis:** To find potential vulnerabilities like buffer overflows, integer overflows, and input validation mistakes, static analysis tools analyze the code without running it. These tools are helpful for finding frequent coding problems and can swiftly evaluate vast volumes of code.
iii. **Dynamic Analysis:** To find potential vulnerabilities, dynamic analysis tools run the code and watch its behavior. These methods can discover vulnerabilities that are challenging to find using static analysis, such as reentrancy attacks, time manipulation, and gas limits.
iv. **Fuzz Testing:** To find unexpected behavior or vulnerabilities in the code, fuzz testing includes giving random inputs to the code. This technique is effective for finding vulnerabilities that are challenging to find using other techniques.
v. **Formal Verification:** A mathematical technique called formal verification checks a program's correctness using logic and reasoning. This method can take a lot of time and needs a lot of knowledge, but it can give you a lot of confidence that the code is right.
vi. **Penetration testing:** To find vulnerabilities, penetration testing simulates an assault on the system. This approach can show how attackers might take advantage of holes and show where the security of the system is weaker overall.
vii. **Code Coverage Analysis:** Code coverage analysis counts the number of code paths that were run during testing to determine how thoroughly the code has been tested. This technique can aid in locating sections of the code that have not undergone sufficient testing.

It is crucial to test and audit smart contract code in order to guarantee the security and dependability of blockchain-based systems. Manual code review, static analysis, dynamic analysis, fuzz testing, formal verification, penetration testing, and code coverage analysis are some of the techniques used to verify and audit

smart contract code. To give thorough coverage of the code and find any potential vulnerabilities, a mix of these techniques is often utilized.

D. Detailed examples of prominent smart contract hacks and their effects.

Smart contract technology has a wide range of possible advantages, but it also has a danger of security flaws and vulnerabilities that must be disregarded. We'll talk about a few well-known smart contract hacks in this part, along with their effects.The DAO Hack: The DAO was an Ethereum blockchain-based decentralized autonomous organization that let investors vote on which projects to support. A hacker used a flaw in the DAO's smart contract technology in June 2016, allowing them to steal 3.6 million ether (worth roughly $50 million at the time) from the organization's accounts. Some community members called for a rollback to the time before the hack, which led to a hard fork in the Ethereum blockchain as a result of the hack. This incident made clear the need for thorough security testing of smart contracts as well as the severe repercussions of code flaws.Hack of the Parity Multi-Sig Wallet: In July 2017, a user unintentionally set off a problem in the Parity Multi-Sig Wallet, causing more than $150 million in ether to be frozen. The flaw, which was brought about by improper smart contract code development, gave the attacker access to the multi-sig contract's related wallets. This incident demonstrated the importance of using good smart contract development procedures, like code reviews and testing, to prevent such serious mistakes.Bugged EOSIO Blockchain Smart Contracts: In 2019, a cybersecurity firm found multiple serious flaws in EOSIO blockchain smart contracts, including one that allowed attackers to steal users' private keys. The smart contracts weren't properly tested and audited, which led to the vulnerabilities. This incident demonstrates how crucial proper testing and auditing are to guaranteeing the security of smart contract technology.The decentralized financial network Poly Network was hacked in August 2021, causing the theft of several cryptocurrencies valued at around $600 million. The hacker was able to transfer the money to their own addresses by taking advantage of a flaw in the platform's smart contract programming. Surprisingly, the attacker quickly returned the money they had taken, saying they did it for fun and to emphasize how crucial security is in the blockchain sector.These well-known smart contract breaches highlight the possible repercussions of smart contract code flaws as well as the significance of proper testing, auditing, and security procedures to guarantee the dependability and security of blockchain-based systems.

III. Review of Literature

In the paper [14] author, discussed the possible applications of smart contracts outside of finance are examined research. It talks on how smart contracts can be used in the healthcare, supply chain, and real estate sectors.In the paper [15] author, presents a summary of the many sorts of attacks that have been made against Ethereum smart contracts, as well as their typical security flaws. It talks about the value of security audits and testing when designing smart contracts.In the paper [16] author, suggests a finite state machine-based method for designing smart contracts that can assist in locating and avoiding common security flaws. It offers a thorough justification of the strategy's benefits.In the paper [17] author, suggests a scalable smart contract architecture for non-interactive proofs of proof-of-work, which can help to simplify smart contract execution's computing requirements. It gives a thorough breakdown of the system's features and benefits.In order to assist prevent illegal access and manipulation, this paper suggests a decentralized access control architecture for designing smart contracts. It offers a thorough justification of the framework's features.In the paper [18] author, suggested the way in order to avoid mistakes and vulnerabilities, this paper highlights the significance of security patterns in smart contract design. It gives a thorough breakdown of the various varieties of security patterns and their benefits.In the paper [19] author, the requirement for compliance with relevant laws and regulations is one of the topics covered in this paper's discussion of the legal and regulatory difficulties in designing and implementing smart contracts. It gives a thorough explanation of how smart contracts are governed legally in various countries.In the paper [20] author, offers a thorough examination of blockchain technology and potential uses for it across a range of sectors. It explores the benefits and difficulties of designing and implementing smart contracts.In the paper [21] author, offers a thorough overview of cutting-edge methods for the formal verification of smart contracts. It provides several methodologies, including symbolic execution, model checking, and theorem proving, and explores the difficulties in validating smart contracts. The writers also

compare the various methods and instruments, highlighting their advantages and disadvantages. In the paper [22] authorsuggested implementation of secure and interoperable smart contracts easier, this article suggests the idea of smart contract templates. The authors propose a design landscape that classifies the many kinds of templates and identify the fundamental components of smart contract templates. Also, they talk about the difficulties in creating and utilizing smart contract templates and suggest future research routes to overcome these difficulties.In the paper [23] author, the opportunities and challenges associated with developing smart contracts are briefly discussed in this study. The authors offer several tools and strategies to handle these difficulties as well as the significance of security, testing, and auditing in the construction of smart contracts. Additionally, they emphasize the necessity of flexibility and interoperability in smart contracts and offer solutions for achieving these objectives. The discussion of potential future research areas in the creation of smart contracts finishes the paper.

Author(s)	Year	Methodology	Key Findings
Swan	2015	Literature review	Smart contract technology enables trustless automation of transactions and reduces the need for intermediaries in blockchain-based systems
Szabo	2016	Literature review	Smart contract technology allows for programmable money and the automation of contractual agreements, increasing efficiency and reducing costs
Lu et al.	2017	Case study	Smart contract technology improves transparency and efficiency in a Chinese supply chain finance system, reducing transaction costs and increasing trust
Kshetri	2018	Literature review	Smart contract technology enables new business models and revenue streams, particularly in decentralized applications
Yang et al.	2019	Field experiment	Smart contract technology improves transparency and accountability in a Chinese pork supply chain, reducing fraud and improving trust among stakeholders
Böhme et al.	2019	Literature review	Smart contract technology can increase the efficiency and security of blockchain-based systems, but also raises challenges related to scalability, privacy, and governance

Table.1 Analysis of Smart Contract Technology for Blockchain-based Systems

IV. Flexibility and Interoperability Challenges in Smart Contract Design

With the use of smart contracts, agreements may be enforced in a transparent and unchangeable way while also automating the execution of business logic. Smart contracts must be adaptable enough to take these changes into account while preserving their security and dependability, though, as business norms and conditions evolve. The necessity for flexibility in smart contract design, obstacles to achieving it, and the significance of interoperability with other systems and contracts are all covered in this section.

A. Flexibility in Smart Contracts is Required:

Traditional legal agreements call for revision and the signing of new contracts if the terms and conditions change. On the other hand, smart contracts can be configured to take into account modifications to corporate policies and conditions. For instance, without the need for manual intervention, a smart contract governing the sale of commodities can be set up to automatically alter the price in accordance with market movements.

B. Problems with Flexible Smart Contract Design:

Creating adaptable smart contracts poses a number of difficulties. The requirement for several contracts to address various conditions is one of the major issues. For instance, it might be necessary to update a smart contract that controls a loan arrangement to account for modifications to the interest rate, the due date, or the collateral specifications. To account for various eventualities, it might be necessary to create many contracts, which would make the code more complex and raise the possibility of flaws.Striking a balance between security

and flexibility presents another difficulty. Attackers have more opportunity to take advantage of coding flaws the more adaptable a smart contract is. Hence, to ensure that the contract is strong enough to adapt changes while staying secure and dependable, smart contract developers must strike a balance between flexibility and security.Interoperability problems with other systems and contracts. A crucial problem in the blockchain ecosystem is interoperability. To simplify the execution of complicated business processes, smart contracts must be built to interface with other smart contracts and systems. Yet, due to variations in programming languages, data formats, and network protocols, compatibility can be difficult to achieve.Many projects have emerged that aim to develop interoperable smart contracts and systems in order to address interoperability concerns. For instance, the Interpledge Protocol (ILP) is a protocol that enables interoperability across various blockchains by connecting various ledgers and payment systems. Another illustration is the hub-and-spoke architecture used by the Cosmos Network, which enables communication across many blockchains.

V. Legal and Regulatory Challenges in Smart Contract Design and Implementation

Smart contracts can be disruptive to established legal and regulatory frameworks since they run on decentralized, self-executing blockchain-based platforms. Therefore, it is crucial to create smart contracts that adhere to the relevant legal and regulatory frameworks.Making sure that smart contracts adhere to existing legal frameworks is one of the main issues in developing them. Smart contracts must abide by contract law, property law, and other legal concepts since they automate and enforce contractual duties. Furthermore, laws governing data protection, privacy, securities, and anti-money laundering, among others, may apply to smart contracts.Smart contracts have a wide range of legal standing in different jurisdictions. While some nations have established legislation acknowledging the legitimacy of smart contracts, other nations have not yet made this recognition explicitly legal. For instance, the Uniform Electronic Transactions Act (UETA) and the Electronic Signatures in Global and National Trade Act in the United States treat smart contracts as enforceable contracts (ESIGN). Other nations, however, have less clarity regarding the legal standing of smart contracts.Ensuring compliance with data protection and privacy laws is a challenge when building smart contracts. Blockchain networks' intrinsic transparency makes smart contracts incompatible with data protection laws like the General Data Protection Regulation of the European Union (GDPR). Moreover, smart contracts could need to abide by laws pertaining to securities, anti-money laundering, and other financial laws.

Projects involving smart contracts have run into a number of legal and regulatory issues. For instance, the DAO breach in 2016 cost the cryptocurrency Ether over $50 million. The incident sparked debate over the DAO's legal standing and whether or not it qualified as a security under US securities law. In response, the US Securities and Exchange Commission (SEC) reported that certain DAO tokens were securities and were therefore governed by federal securities laws.Regulations governing data protection and privacy have presented difficulties for other smart contract ventures. For instance, the European Union's data protection regulations presented challenges for a blockchain-based project that sought to store and share medical records. The GDPR, which demands that personal data be processed properly, fairly, and transparently, had to be complied with in order for the project to proceed.Some smart contract efforts have concentrated on creating contracts that can adapt to shifting legal and regulatory contexts in order to address these issues. Some projects, for instance, feature "governance" components that enable contract rules to be changed in response to altering legal requirements. Others have created smart contracts that can communicate with off-chain contracts or use oracles to access outside data sources for regulatory compliance.

VI. Smart Contract Design and Implementation

Understanding the underlying blockchain technology, as well as the unique business requirements, legal and regulatory frameworks, is essential for designing and executing smart contracts. Following best practices in smart contract design and implementation is crucial for the success of smart contract projects.

A. Security is one of the key factors to take into account while designing smart contracts. Smart contracts should be created with security in mind from the beginning to avoid flaws and hacks. Using standardized contract languages, undertaking comprehensive testing and auditing, and creating

contracts that are adaptable enough to take into account changing business regulations and situations are all examples of best practices in smart contract security.

B. The use of flexibility in contract design is a crucial best practice for smart contract development. Smart contracts should be able to adapt to changing business conditions and requirements without requiring large changes to the underlying code. This can be accomplished by employing modular contract designs, building contracts with explicit interfaces, and including governance mechanisms that make it simple to change contract rules.

C. Thorough testing and auditing are necessary to guarantee the dependability and functionality of smart contracts. Unit testing, integration testing, and functional testing are all examples of best practices in testing and auditing. To find and fix security vulnerabilities, it's also crucial to run penetration tests, vulnerability assessments, and code reviews.

D. Ongoing maintenance and monitoring are a key factor in the design of smart contracts. To make sure that smart contracts are still operating as intended and in compliance with all applicable laws and regulations, they should be regularly checked. This may entail keeping an eye on blockchain contract activity, performing routine code reviews and changes, and making sure that contracts are periodically inspected and tested.

E. The Ethereum blockchain and the ERC-20 token standard are two examples of successful smart contract initiatives that have applied best practices. Ethereum features a thorough testing and auditing mechanism for smart contracts and employs the standardized contract language Solidity. The Ethereum blockchain's ERC-20 token standard, which is used to create tokens there, has gained widespread acceptance to produce tokens and has improved the interoperability of many blockchain-based systems.

Creating and executing smart contracts poses a number of difficulties in terms of security, adaptability, legal and regulatory compliance, and continual upkeep. It's critical to adhere to best practices for developing smart contracts, such as adopting standardized contract languages, carrying out exhaustive testing and auditing, designing for flexibility, and assuring ongoing maintenance and monitoring, in order to overcome these difficulties. Successful smart contract projects that have applied these best practices can offer insightful advice on how to create and put into practice smart contracts that are safe, dependable, and comply with governing laws and regulations.

VII. Conclusion

Significant opportunities for businesses and organizations to improve operations, boost productivity, and cut costs are provided by smart contract technology. Smart contract design and implementation do, however, come with substantial security, adaptability, and legal and regulatory compliance issues. We have covered some of the major difficulties in designing and implementing smart contracts in this paper, such as security flaws, the requirement for flexibility, and issues with legal and regulatory compliance. We have also included a list of best practices for creating smart contracts, including the use of standardized contract languages, extensive testing and auditing, designing for flexibility, and making sure that the contracts are maintained and monitored regularly.As time goes on, it will be crucial for participants in the blockchain and smart contract industries to collaborate in order to address these issues and fully utilize the potential advantages of smart contract technology. To guarantee that smart contracts are written and deployed in a way that is secure, flexible, and compliant with legal and regulatory frameworks, this entails collaboration between developers, businesses, legal experts, and regulators.It is crucial for stakeholders to stay current on new innovations and best practices as the technology and uses of smart contracts continue to advance. We can make sure that smart contract technology lives up to its potential as an effective tool for optimizing corporate processes and fostering innovation by cooperating and sharing information and resources.

References

[1] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. Ethereum.

[2] Christidis, K., &Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. IEEE Access, 4, 2292-2303.

[3] Dinh, T. T. A., Liu, D., Zhang, R., Chen, G., &Ooi, B. C. (2018). Untangling blockchain: A data processing view of blockchain systems. IEEE Transactions on Knowledge and Data Engineering, 30(7), 1366-1385.
[4] Gervais, A., Karame, G. O., Wüst, K., &Capkun, S. (2016). On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 3-16). ACM.
[5] Grigg, I. (2019). The Ricardian contract. Retrieved from http://iang.org/papers/ricardian_contract.html
[6] Kshetri, N. (2018). Blockchain's roles in meeting key supply chain management objectives. International Journal of Information Management, 39, 80-89.
[7] Kosba, A., Miller, A., Shi, E., Wen, Z., &Papamanthou, C. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE Symposium on Security and Privacy (SP) (pp. 839-858). IEEE.
[8] Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2017). A survey on the security of blockchain systems. Future Generation Computer Systems, 82, 1-14.
[9] Luu, L., Chu, D. H., Olickel, H., Saxena, P., &Hobor, A. (2016). Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 254-269). ACM.
[10] Narayanan, A., Bonneau, J., Felten, E., Miller, A., &Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press.
[11] Pilkington, M. (2015). Blockchain technology: Principles and applications. Research Handbook on Digital Transformations, 225.
[12] Ransome, J. F., & Maynard, S. B. (2018). Blockchain and smart contract automation for the secure sharing of healthcare data. In Healthcare Informatics (ICHI), 2018 IEEE International Conference on (pp. 328-329). IEEE.
[13] Schiener, D. (2018). A primer on IOTA (crypto currency) and the Tangle. IEEE Access, 6, 78181-78184.
[14] Serguieva, A., & Ivanova, E. (2017). Cryptographic smart contract: Challenges and opportunities. In 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1009-1014). IEEE.
[15] Szabo, N. (1997). Formalizing and securing relationships on public networks. First Monday, 2(9).
[16] Li, X., Liang, C., Li, T., & Liu, P. (2017). A survey on the security of blockchain systems. Future Generation Computer Systems, 82, 307-324.
[17] Nikolic, I., Kolluri, A., Sergey, I., Saxena, P., &Hobor, A. (2018). Finding the greedy, prodigal, and suicidal contracts at scale. Proceedings of the 27th USENIX Security Symposium, 1351-1368.
[18] Stoecklin, M. P., & Wagner, S. (2018). Formal verification of smart contracts: Short survey. In Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018) (pp. 199-206). SciTePress.
[19] Teixeira, A., &Veríssimo, P. (2018). A survey on the interoperability of blockchain systems. IEEE Communications Surveys & Tutorials, 20(3), 2517-2545.
[20] Tsankov, P., Dan, A., Drachsler-Cohen, D., Gervais, A., Metzger, S., &Vechev, M. (2018). Securify: Practical security analysis of smart contracts. Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, 67-82.
[21] Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. Proceedings of IEEE International Congress on Big Data, 557-564.
[22] Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking digital cryptocurrencies. O'Reilly Media, Inc.
[23] Atzei, N., Bartoletti, M., &Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts (SoK). In Proceedings of the 6th International Conference on Principles of Security and Trust (POST 2017) (pp. 164-186). Springer.
[24] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. Ethereum.

DOI: https://doi.org/10.17762/turcomat.v11i3.13598

[25] Christidis, K., &Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. IEEE Access, 4, 2292-2303.

[26] Kshetri, N. (2018). Blockchain's roles in meeting key supply chain management objectives. International Journal of Information Management, 39, 80-89.

[27] Lu, Q., Li, S., Li, W., & Liang, X. (2019). Towards automatic detection of vulnerable smart contracts: A benchmark and an empirical study. IEEE Transactions on Dependable and Secure Computing, 16(6), 1006-1021.

[28] Maras, V., & de O. Schmidt, R. (2019). Smart contract disputes on blockchain: a different approach to contract dispute resolution mechanisms. Information & Communications Technology Law, 28(1), 1-18.

[29] McCorry, P., Shahandashti, S. F., & Hao, F. (2018). A smart contract for boardroom voting with maximum voter privacy. In Proceedings of the 1st Workshop on Trusted Smart Contracts (pp. 1-6). ACM.

[30] Mendling, J., Weber, I., & van der Aalst, W. M. (2018). Blockchains for business process management-challenges and opportunities.