# The Potential And Security Of Smart Contracts.

1st Prof. Sumit Shevtekar
Asst.Professor
*Department of Computer Engineering*
*Pune Institute of Computer Technology*
Pune, India

2nd Sushant Langhi
Student
*Department of Computer Engineering*
*Pune Institute of Computer Technology*
Pune, India

***Abstract*****—**In this paper, we investigate the potential and security of running smart contracts based on an open distributed network like those of cryptocurrencies. We will introduce our- selves to the blockchain technology and Smart Contracts on them. Exploring the applications, security and proposed security enhancements and recent technical advances

***Index Terms*****—**Blockchain, Smart Contracts, Security,Potential, Recent Technical Advances

## I. Introduction.

Blockchain stands as a brand new technology that has reshaped our perspectives on data storage, security, and trust in the modern digital era. It was initially introduced by an anonymous author, Satoshi Nakamoto, who developed Bitcoin to transfer digital currencies directly without the need for third parties. Fundamentally, a blockchain serves as a digital ledger that operates in a decentralized and distributed manner. Each block records transactions across a network of computers, ensuring transparency, security, and immutability.

Smart contracts are a digital innovation that streamlines business agreements. They run on computers independently, following preset rules. Think of them as digital assistants that ensure agreements happen smoothly. Understanding smart contracts is important as they redefine how businesses operate. They are like a peek into the future, promising improved efficiency and security in agreements. We will explore their potential to make business more efficient and their critical aspect of security. Smart contracts, as a concept, were not invented by a single individual; rather, they evolved over time as a part of the broader development of blockchain technology. However, the term ”smart contract” was popularized by computer scientist and legal scholar Nick Szabo in the 1990s.

## II. Key Concepts to Understand.

### *A. Blockchain Technology.*

Blockchain technology is a ledger system that operates in a distributed and decentralized manner. Its purpose is to securely and transparently record transactions across a network of computers in an immutable way. The ’blocks’ are pieces of information about these transactions. Example for explanation

Imagine you have a digital currency called ’CryptoCoin’ and you want to transfer some CryptoCoins to your friend Alice. Instead of using a traditional bank, you decide to use a blockchain to make this transfer. You initiate a transaction by creating a record of the transfer. This transaction contains information about the sender (you), the recipient (Alice), the amount of cryptocoins to be transferred, and a digital signature to prove that you authorized the transfer.

Your transaction is broadcast to a network of computers, often referred to as nodes or miners. These nodes validate your transaction to ensure it meets specific criteria. They check things like whether you have enough cryptocoins in your account to make the transfer and that your digital signature is valid. These transactions in our example are the blocks that form a chain of information; a copy of the entire chain is what is shared by everyone on the network.

### *B. Smart Contracts.*

Smart contracts are a form of digital agreement that executes autonomously based on predefined conditions coded in software using fields and functions. Unlike traditional legal contracts, smart contracts replace human intermediaries and rely on blockchain technology for their execution and enforcement. These digital contracts are designed to offer transparency, security, and automation in various domains. They are programmed to execute automatically when specific conditions or triggers are met. These conditions are typically based on external data sources or user inputs. Trust and transparency are ensured by their execution on blockchain networks. All participants in the network can verify the execution, and the results are recorded in an immutable ledger, which is the blockchain.

### *C. Platforms and Consensus Mechanisms.*

Consensus mechanisms determine the validation of blocks in a particular blockchain. There are several consensus mechanisms that are used by different platforms for validation. They are categorized into two types:

- Voting based
- Proof based

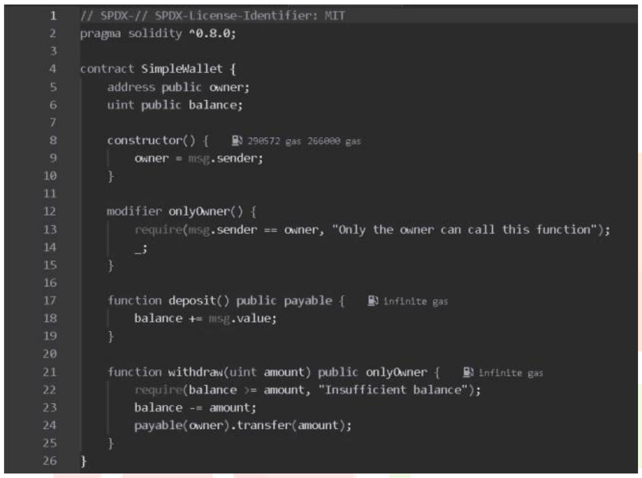
In proof-based mechanisms, a head is selected to validate and append new blocks to the blockchain, while in voting-based mechanisms, multiple nodes vote for the validation of a block, and on the basis of a consensus mechanism used by that particular blockchain, the block is validated.

The most widely applied consensus mechanisms are as follows:

1) Proof of Work (PoW): It is an incentive-based mechanism wherein the various nodes (called miners) solve complicated mathematical puzzles to earn rewards for their work.
2) Proof of Stake (PoS): Introduced to reduce the cost of the PoW mechanism, a miner is chosen based on the miner's stake value. The miner proposes a block, which is then validated by other miners. The miner proposing the block gets rewarded.
3) Delegated Proof of Stake (DPoS): This is PoS, but on a smaller scale. A subset of miners is selected by stakeholders at random, who then validate and append the block. Since the validating nodes are fewer, this is more efficient.

There are several such mechanisms that overcome the drawbacks of each other.

### D. Smart Contract Example



```
// SPDX-// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleWallet {
    address public owner;
    uint public balance;

    constructor() {    290572 gas 266000 gas
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can call this function");
        _;
    }

    function deposit() public payable {    infinite gas
        balance += msg.value;
    }

    function withdraw(uint amount) public onlyOwner {    infinite gas
        require(balance >= amount, "Insufficient balance");
        balance -= amount;
        payable(owner).transfer(amount);
    }
}
```

This example, the ”SimpleWallet” smart contract, is an example of a digital wallet written in Solidity. It allows the contract owner to deposit and withdraw Ether securely. The contract initializes with the owner's address and tracks the wallet's balance. The ”onlyOwner” modifier ensures that critical functions can only be executed by the contract owner, preventing unauthorized access. Users can deposit Ether into the wallet using the ”deposit” function, which adds the deposited amount to the wallet's balance. On the other hand, the ”withdraw” function allows the owner to withdraw a specified amount of Ether, but only if the wallet contains sufficient funds. This simple contract demonstrates the use of modifiers, conditional checks, and basic wallet operations on the Ethereum blockchain.

## III. Smart Contract Potential.

The potential of smart contracts lies in their own creation and working. The potential properties of smart contracts are:

- Security:
  Security is the top priority of a smart contract, and it is achieved through a combination of cryptographic techniques and decentralized networks. These cryptographic techniques ensure that these completely automated agreements are encrypted and can only be accessed by authorized parties with the appropriate cryptographic keys.
- Transparency:
  All smart contracts are blocks of code that are stored in a blockchain that is open-source, meaning they can be understood, reviewed, and accessed by anyone on the network. This transparency is a major factor in developing an understanding of a smart contract.
- Immutability:
  Once deployed on a blockchain, these self-executing agreements are unchangeable and permanent. This immutability ensures that the terms and conditions of the contract remain fixed and transparent throughout their lifecycle, as they are encoded in the contract's code.

Moreover, the transparent and automated nature of these digital agreements builds trust by ensuring that the terms and outcomes are clear and resistant to manipulation. Users can rely on the code and blockchain technology to execute fair and trustworthy transactions. Simultaneously, smart contracts enhance efficiency by automating processes, reducing the need for intermediaries, and minimizing the potential for human error. This approach not only saves time but also lowers costs, making smart contracts a reliable and efficient solution for a wide range of applications, from financial transactions to supply chain management.

### A. Finance.

Banks and other financial institutions are governed by a centralized authority. All transactions that occur in such institutions are carried out or overlooked by intermediaries. Smart contracts give a revolutionary turn to these automated agreements, powered by blockchain technology, streamlining processes, reducing the risk of errors, and significantly cutting operational costs. In decentralized finance (DeFi), smart contracts are at the center of lending and borrowing platforms, decentralized exchanges, and yield farming protocols, providing users with entire control over their assets while eliminating the need for such intermediaries [1].

*1) Decentralized Lending and Borrowing:* Blockchain platforms like Compound and Aave use smart contracts to allow users to lend and borrow cryptocurrencies without the need for traditional banks. Borrowers provide collateral, and smart contracts automatically execute loans and interest payments.

For example: A borrower wants to borrow ETH using BTC as collateral. The borrower deposits their BTC into a smart contract pool. The borrower specifies the amount of ETH they want to borrow and the interest rate they are willing to pay. A lender deposits ETH into the smart contract pool. The smart contract matches the borrower and lender, and the borrower receives the ETH loan. The borrower begins paying interest on the loan. Once the loan is repaid, the borrower's BTC collateral is released back to them. Thus, the borrower can trade on platforms based on the Ethereum blockchain, opening up new possibilities.

Decentralized exchanges (DEXs) work in a similar fashion, except there is no lending or borrowing. Two peers might want to exchange a currency or cryptocurrency, and they can do so using such DEX smart contracts.

*2) Tokenized Assets:* Real estate and art can be tokenized using smart contracts, making it easier for investors to own fractions of high-value assets, basically facilitating the representation of real-world assets on a blockchain.

For example: Only the owner of the real estate can issue tokens representing ownership of his or her property. Each token represents a specific fraction of the ownership of the property. Tokens can only be transferred between verified buyers and sellers. The smart contract will automatically distribute rent payments to token holders based on their ownership stake.

Assets, as such, can be anything ranging from real estate and fine art to actual intellectual properties. Real estate properties like the St. Regis Aspen Resort were tokenized, allowing investors to purchase fractional ownership in the resort. Non-fungible tokens like the Bored Ape Yacht Club collection, which is an arguable fine art collection.

*3) Insurance Claims and Payouts:* Insurance companies are exploring smart contracts to automate the claims and payout processes. When predefined conditions are met (e.g., flight delays or weather events), smart contracts can automatically trigger insurance payouts.

For example: A policyholder files a claim using a dedicated portal or mobile app. The smart contract gathers all relevant data related to the claim, such as the policyholder's information, the type of claim, and the amount of the claim. The smart contract verifies the claim by checking the policyholder's policy and other relevant information. If the claim is valid, the smart contract automatically initiates a payout. The payout is sent to the policyholder's crypto wallet or bank account, depending on their preference, without intermediaries.

### *B. Healthcare.*

The healthcare industry relies on the medical reports of their patients to further carry out procedures. But doctors sometimes suffer from inconvenient communication about their patients health conditions from the past. Smart contracts can help bridge this miscommunication by:

*1) Medical records management:* Smart contracts can be used to create and manage electronic medical records (EMRs) in a secure and transparent way. This would give patients control over their own medical data and allow them to share it with healthcare providers on a need-to-know basis. Smart contracts can also be used to automate the process of updating and transferring medical records, which can save time and reduce errors.

For example: A patient creates a new medical record using a dedicated portal or mobile app. The smart contract encrypts the patient's medical record and stores it on the blockchain. The patient can then share their medical record with healthcare providers on a need-to-know basis. When a healthcare provider accesses the patient's medical record, the smart contract verifies their identity and ensures that they have the appropriate permissions. The healthcare provider can then view the patient's medical record and make any necessary updates. All updates to the patient's medical record are recorded on the blockchain and are visible to the patient.

*2) Drug supply chain management:* Smart contracts can be used to track the movement of drugs through the supply chain. It can be used to create a digital record of each drug, including its origin, destination, and all of the parties involved in its transportation and storage. This information can then be used to track the drug's movement in real time and identify any potential problems, such as counterfeit drugs or delays in delivery. This can help to ensure the quality and safety of drugs and can also help to identify and prevent counterfeit drugs. Drug supply chain management smart contracts are still in their early stages of development, but they have the potential to revolutionize the pharmaceutical industry by making the supply chain more efficient, transparent, and secure.

The healthcare industry can significantly become more efficient and trusted [1], [2].

## IV. Smart Contract Security.

The security of smart contracts is of great concern in blockchain technology, as these automated digital agreements often handle valuable assets and transactions. Ensuring the security of smart contracts involves several key considerations. First, it's essential to write robust, well-audited code to prevent vulnerabilities that malicious actors could exploit. Code vulnerabilities, if left unaddressed, can result in financial losses and even the loss of assets. Second, secure access control mechanisms must be implemented to restrict interactions with the contract, ensuring that only authorized parties can invoke specific functions. Third, managing the financial resources and ensuring funds are protected against unauthorized access are vital.

The security issues and the recent technical advances made to tackle them can be explained using the levels of their implementation.

1) Level 1: Programming Language Level
   In the initial stage of the creation of a smart contract, we come across issues concerning code readability and code correctness.
   - Code Readability:
     Readability is critical for understanding and maintaining the contract code. Writing code for smart contracts is complex and requires a deep understanding of blockchain technology. Complex code should be simplified, and better comments and documentation are essential for clarity. Though improved development tools, programming languages, and frameworks, like Solidity, have been developed to enhance code readability, it still lacks some effort. Several proposals have been made to implement human-readable code and execution. Frantz and

Nowostawski introduced a semi-automated translation system designed to transform human-readable contract representations into executable computational programs [6].

- Code correctness:
  Developers employ rigorous testing, code audits, and formal verification methods to identify and rectify potential issues before deploying the contract on the blockchain. Comprehensive testing, including both unit and integration testing, is essential to catch bugs and vulnerabilities before deployment. Additionally, code audits by experts in blockchain security can provide an independent assessment of a contract's correctness. Bytecode-level analysis only requires the compiled bytecodes of smart contracts, which are easier to obtain. Several proposals for tools guaranteeing the elimination of bugs by bytecode analysis are being developed [3].

2) Level 2: Virtual Machine Level
The deployment of a validated smart contract is another crucial stage. These contracts being immutable is a boon as criminal exploitation of them is not possible, though any bugs or exceptions on the smart contract can lead to taking the entire contract down to fix them as they are immutable.

- Re-entrancy:
  Functional issues center around preventing vulnerabilities and bugs. Re-entrancy refers to a vulnerability where an external contract can repeatedly call back into the current contract before the initial call completes, potentially leading to unintended and malicious behaviors. Proper access control mechanisms should be implemented to restrict unauthorized access, and measures must be in place to prevent reentrancy attacks and ensure that external calls are carefully managed. Recommendations to eliminate re-entrancy vulnerabilities by prohibiting nesting calling among functions in the contract have been made, and proposals to perform fuzz testing on smart contracts by iteratively generating random but diverse transactions to detect reentrancy bugs have also been introduced. [7], [8]
- Dynamic flow control:
  Smart contracts on blockchains must operate deterministically and predictably across all nodes in the network. Dynamic control flow, such as loops and conditionals, can introduce unpredictability, which can be a security concern. Developers should consider potential leads to issues like gas limit exhaustion. To address this, blockchain platforms and programming languages like Solidity have introduced gas limits, which restrict the computational resources a contract can consume.

3) Level 3: Blockchain Level
The rightful execution of the smart contract determines its final state. The contract will be broadcasted to other miners (nodes) after deployment on the blockchain, where it will modify the state of the system. A trustworthy oracle, better transaction ordering, and security from scams and frauds are the main concerns at this stage.

- Trustworthy oracle:
  Smart contracts need information about the real world to operate. Oracles act as bridges between the blockchain and off-chain data sources, that is, the real-world data sources, providing smart contracts with the information necessary for decision-making and execution. However, ensuring the reliability of oracles is a significant challenge, as they introduce a potential single point of failure and may be susceptible to manipulation or data inaccuracies. A smart contract-based solution tackling Oracle reliability has been introduced. A reputation record is used to provide real-life data on the oracle's performance, and the oracle with the highest rating of reliability is finalized.
- Scams and Frauds:
  While blockchain technology offers transparency and security, it does not inherently prevent malicious actors from deploying fraudulent smart contracts or deceiving users. Smart contracts are prone to malicious attacks by criminal-minded individuals. Finding vulnerabilities that can be exploited to gain financial assets has endangered the true capabilities of smart contracts. To counter such advances, users should thoroughly research projects and contracts before participating, verify the identities of project teams, and have knowledge of anything that appears too good to be true. Honeypots are an advancement used to tackle such behavior. These are vulnerable-looking contracts that contain hidden traps tricking the individual with malicious intent to gain nothing but failure while trying to exploit the vulnerability.

There are many more issues contained deeper in each of these levels that harm the security of smart contracts. A developer should work tirelessly to identify and shut down all points of failure in a smart contract.

## V. Conclusion and Future Scope.

The world is still navigating regulatory and security challenges in this transformative landscape, trying hard to gain a balance between innovation and safeguarding against potential risks. As smart contracts continue to evolve, their role in reshaping the financial, healthcare, industrial, and other important sectors is undeniable, offering both opportunities and challenges for the future.

## Acknowledgments

I am truly grateful for the valuable learning experience gained while preparing this report on ”The Potential and Security of Smart Contracts.” My heartfelt appreciation goes out to our seminar coordinator, Prof. D.D. Kadam, the head of the department, Dr. Geetanjali Kale, and the Principal Dr. S.T. Gandhe for their unwavering support and encouragement.

I would like to extend my sincere gratitude to my mentor, Prof. S.S. Shevtekar from the Department of Computer Science, for his valuable guidance and assistance. His consistent encouragement played a pivotal role in the successful completion of this seminar project.

## References

[1] Zheng, Zibin, et al. ”An overview on smart contracts: Challenges, advances and platforms.” Future Generation Computer Systems 105 (2020): 475-491.

[2] Wang, Zeli, et al. ”Ethereum smart contract security research: survey and future research opportunities.” Frontiers of Computer Science 15 (2021): 1-18.

[3] Rouhani, Sara, and Ralph Deters. ”Security, performance, and applications of smart contracts: A systematic survey.” IEEE Access 7 (2019): 50759-50779

[4] Luu, Loi, et al. ”Making smart contracts smarter.” Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016.

[5] Michael Coblenz. Obsidian: A safer blockchain programming language. In Proceedings of the 39th International Conference on Software Engineering Companion, ICSE-C ’17

[6] Christopher K Frantz and Mariusz Nowostawski. From institutions to code: towards automated generation of smart contracts. In Proceedings of IEEE International Workshops on Foundations and Applications of Self Systems, pages 210–215, 2016.

[7] Anastasia Mavridou and Aron Laszka. Tool demonstration: Fsolidm for designing secure ethereum smart contracts. In International Conference on Principles of Security and Trust, pages 270–277. Springer, 2018.

[8] Chao Liu, Han Liu, Zhao Cao, Zhong Chen, Bangdao Chen, and Bill Roscoe. Reguard: finding reentrancy bugs in smart contracts. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, pages 65–68. ACM, 2018.