STM JOURNALS

Journal of
Open Source Developments

ISSN: 2395-6704
Volume 13, Issue 1, 2026
January–April
DOI (Journal): 10.37591/JoOSD

https://journals.stmjournals.com/joosd

Review JoOSD

Smart Contract: A New Buzz Word for Reliable Decentralized Network

Shyamsunder Manaktala[1,*], Rakesh Saxena[2], Deepak Sankhla[3], Aashish Kumar Sharma[3], Rakesh Kumar Kardam[3]

Abstract

Blockchain technology has emerged as a foundational distributed-ledger architecture enabling decentralized, transparent, and tamper-resistant data management across diverse digital ecosystems. This paper presents an integrated review of blockchain fundamentals, consensus mechanisms, smart contract design, and real-world applications beyond cryptocurrencies. It discusses the evolution of smart contracts as automated, self-executing programs that remove intermediaries and ensure deterministic execution, while highlighting critical security vulnerabilities such as reentrancy, arithmetic overflows, and improper access control. This paper judges the need for organized audits, verifies prominent open-source analysis tools—including Slither, Mythril, Securify, and Oyente—and outlines a structured auditing workflow combining static, dynamic, and formal verification techniques. Also, evaluates Broader industry challenges, such as scalability constraints, interoperability gaps, regulatory uncertainty, and the emerging quantum-computing threat, are analyzed to assess the long-term viability of blockchain systems. The study concludes with future research directions emphasizing post-quantum cryptography, cross-chain frameworks, advanced consensus models, and integration of blockchain with AI and IoT to build resilient next-generation decentralized infrastructures.

Keywords: Blockchain, smart contracts, security audits, vulnerability analysis, distributed ledger, consensus mechanisms, decentralized finance (DeFi), post-quantum cryptography, interoperability, Ethereum, static analysis tools, slither, Mythril, scalability, quantum computing threat

INTRODUCTION

Smart contracts have revolutionized blockchain technology by enabling the automation of agreements and transactions beyond the traditional boundaries of cryptocurrency and finance. These contracts are self-executing and programmed with predefined rules that automatically enforce contract terms without requiring intermediaries, which increases trust and reduces operational complexity. Their tamper-resistant nature is invaluable, ensuring that once agreements are recorded, they cannot be altered or disputed retroactively. Smart contracts have facilitated advancements in a diverse range of industries, including governance, real estate, finance, logistics, and supply chains. For example, property transfers in real estate can now be programmatically linked to payment completion, thereby reducing fraud and delays. By using smart contracts, blockchain brings transparency and reliability to everyday transactions. However, the increasing use of smart contracts has raised the demand for robust security practices to avoid critical vulnerabilities [1].

***Author For Correspondence**
Shyamsunder Manaktala
E-mail: ssmanaktala.ece@jecrc.ac.in

[1]Professor, Department of Electronics and Communication Engineering, JECRC University, Jaipur, Rajasthan, India
[2]Professor, Faculty of Computer Engineering, Poornima University, Jaipur, Rajasthan, India
[3]Assistant Professor, Department of Electronics and Communication Engineering, JECRC University, Jaipur, Rajasthan, India

Received Date: February 03, 2026
Accepted Date: April 18, 2026
Published Date: April 30, 2026

Citation: Shyamsunder Manaktala, Rakesh Saxena, Deepak Sankhla, Aashish Kumar Sharma, Rakesh Kumar Kardam. Smart Contract: A New Buzz Word for Reliable Decentralized Network. Journal of Open Source Developments. 2026; 13(1): 35–41p.

Real-World Applications for Blockchain and Smart Contracts and Best Practices for Secure Smart Contract Development

Blockchain protocols, empowered by smart contracts, are widely applied across industries to enhance transparency, traceability, and automation. In supply chain management, each product's journey can be immutably tracked, ensuring authenticity and reducing the chances of counterfeiting or fraud occurring. Decentralized Finance (DeFi) leverages smart contracts to streamline lending, borrowing, and trading, providing financial services without intermediaries and lowering costs, as shown in Figure 1.

In healthcare, sensitive patient data can be securely shared, maintaining both accuracy and privacy. An improved voting system is a good example of a blockchain application, where system security and reliability of the election procedure are maintained, as each vote is a unique transaction; thus, the chances of duplicate voting are tremendously reduced [2].

Intellectual property can be registered and tracked on blockchains, safeguarding rights and preventing unauthorized usage. The rise of blockchain-based gaming and non-fungible tokens (NFTs) demonstrates how the ownership of unique digital items can be protected and exchanged globally. Even energy trading and cross-border payments benefit from blockchain technology by eliminating intermediaries and enhancing efficiency.

Introduction to Smart Contract Security

Smart contracts are self-executing codes that run on blockchain networks. They enable secure, automated, and trustless transactions between parties. However, because they handle valuable assets, security flaws can lead to major financial losses. The document emphasizes that once deployed, smart contracts cannot be easily changed, making vulnerabilities permanent. Therefore, security auditing is essential before deployment [3]. Audits help identify logic errors, coding mistakes, and potential exploitation vectors (Figure 2). The introduction highlights how auditing protects systems, investors, and users alike.

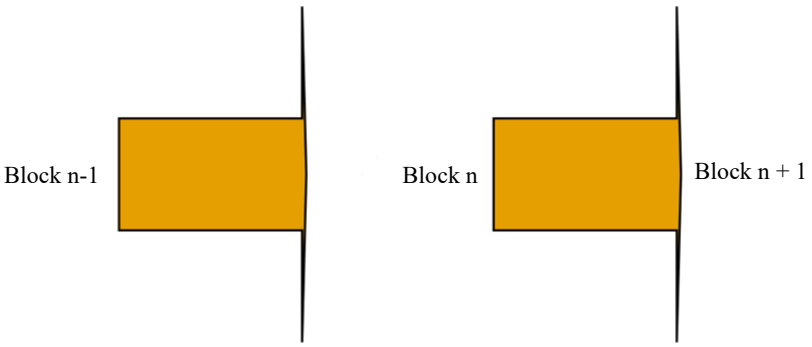

Figure 1. Blockchain block structure.

Figure 2. Smart contract security.

Need for Security Audits and Vulnerability Analysis of Smart Contracts

Despite their promise, smart contracts are vulnerable to programming flaws and logic errors, underscoring the importance of thorough security audits before their deployment. Once deployed, a smart contract's code is immutable, making pre-launch audits the only opportunity to identify vulnerabilities. Flaws in smart contract codes have led to significant financial losses, especially in DeFi and token sales, where hackers exploit weaknesses to siphon funds. The decentralized and permissionless nature of blockchains means that no central regulator can enforce security standards; protection must be baked into the code itself. The increasing complexity of contracts, as they interact with other contracts and off-chain data sources, also raises the risk of unforeseen vulnerabilities. Conventional software testing is inadequate; specialized tools and testing methods are required for these blockchain environments [4]. Regular security audits are fundamental for maintaining trust with users, protecting assets, and safeguarding the reputation and legal standing of blockchain projects, as shown in Figure 3.

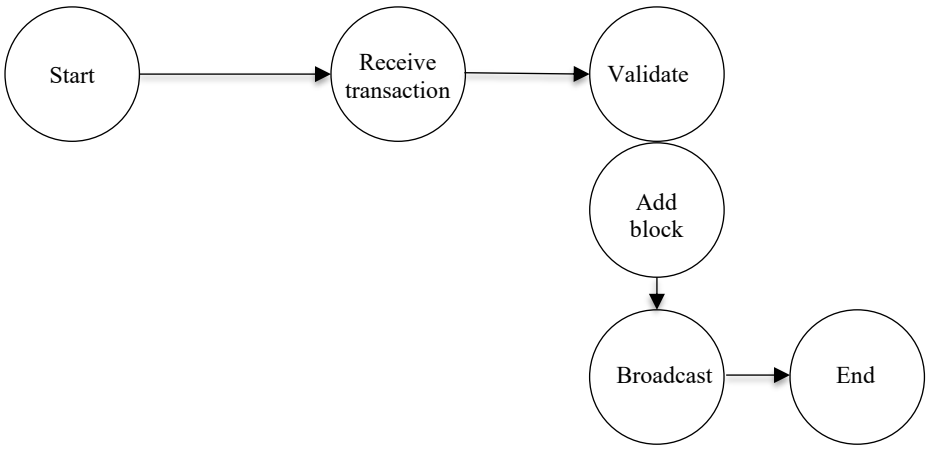

Figure 3. UML activity diagram of blockchain.

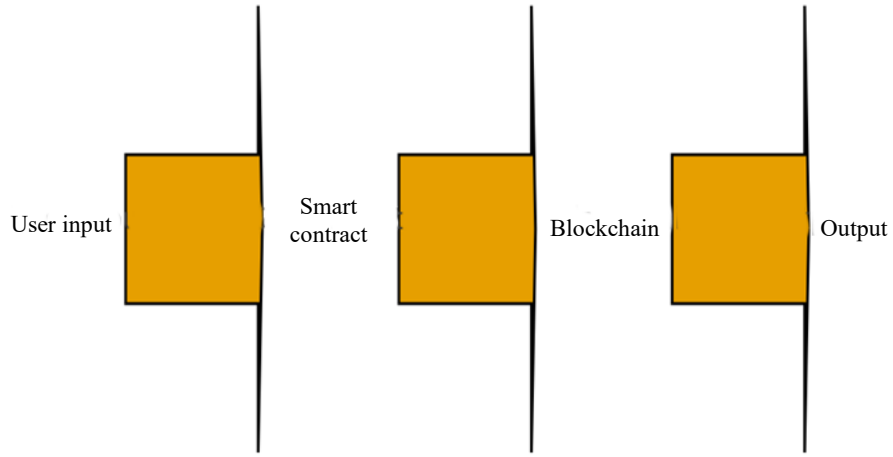

Figure 4. Smart contract execution flow.

Why Security Audits are Necessary

Smart contracts operate without centralized oversight, which means that bugs can be exploited instantly and globally. The document explains that attackers often seek financial gain by exploiting weaknesses in the contract logic. High-profile hacks have demonstrated how vulnerabilities can cause multi-million-dollar losses. Preventative measures, reducing the likelihood of catastrophic system failures. They ensure that contracts behave as intended in all scenarios. Security audits enhance user trust and compliance with industry standards. Thus, auditing is a fundamental requirement rather than an optional enhancement (Figure 4).

Stages of Smart Contract Security Audit

The audit process generally includes code reviews, static analyses, dynamic testing, and manual verifications. A code review ensures human understanding of the contract logic and intent. Static analysis tools scan the code for known patterns of vulnerabilities. Dynamic testing executes smart contracts in simulated environments to detect unexpected behaviors. Manual verification checks the assumptions that automated tools cannot analyze. The document emphasizes thorough documentation at each stage [5]. The final report provides severity ratings and recommendations for remediation (Figure 5).

Common Vulnerabilities in Smart Contracts

Despite their promise, smart contracts are vulnerable to programming flaws and logic errors, underscoring the importance of thorough security audits before they are deployed. Once deployed, a smart contract code is immutable, making pre-launch audits the only opportunity to identify vulnerabilities. Flaws in smart contract codes have led to significant financial losses, especially in DeFi and token sales, where hackers exploit weaknesses to siphon funds. The decentralized and permissionless nature of blockchains means that no central regulator can enforce security standards; protection must be baked into the code itself. The increasing complexity, as contracts interact with other contracts and off-chain data sources, also raises the risk of unforeseen vulnerabilities. Conventional software testing is inadequate, and specialized tools and testing methods are required for these blockchain environments. Regular security audits are fundamental for maintaining trust in users, protecting assets, and safeguarding the reputation and legal standing of blockchain projects [6].

This document discusses typical flaws found in smart contracts, such as reentrancy attacks, overflow/underflow errors, and improper access control. Reentrancy vulnerabilities allow malicious contracts to repeatedly call functions to deplete funds. Arithmetic vulnerabilities can cause unintended financial calculations because of integer overflow. Weak access controls may allow unauthorized users to execute privileged functions. Logic flaws may arise from the incorrect ordering of operations. Poor random sources can allow predictable outcomes in lotteries or games. These vulnerabilities highlight the importance of systematic audit practices (Table 1).

Free and Open-Source Tools for Smart Contract Auditing

Various open-source tools have emerged to detect vulnerabilities and enforce the best coding practices in smart contracts. Tools such as Slither, Mythril, Securify, Oyente, SmartCheck, and Solhint are widely recommended.

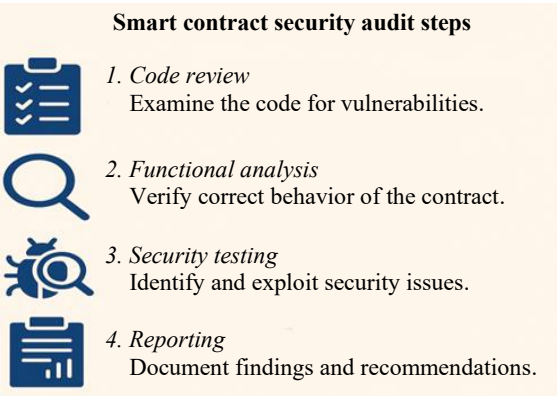

Figure 5. Smart contract security audit steps.

Table 1. Comparison of blockchain features.

Feature	Description	Benefit
Decentralization	No central authority	Eliminates a single point of failure
Immutability	Data cannot be altered	High trust and security
Transparency	Shared ledger across nodes	Accountability

Figure 6. Open-source free smart contract audit tools.

Table 2. Consensus mechanism comparison.

Consensus type	Energy use	Scalability	Security
Proof of work	High	Low	High
Proof of stake	Low	High	Medium
(DPoS)	Low	Very high	Medium
PBFT	Low	Medium	High

DPos, Delegated Proof of Stake; PBFT, Practical Byzantine Fault Tolerance

Slither excels in static analysis and offers detectors for issues such as reentrancy, integer overflows, and uninitialized variables. Mythril combines symbolic execution and static analysis to analyze smart contracts for critical bugs and exploits. Securify and Oyente employ formal methods to verify the correctness of contract behavior and detect typical vulnerabilities. SmartCheck and Solhint scan code and enforce security, coding, and style standards, helping developers to avoid common pitfalls [7]. Using these tools, developers can enhance the safety and reliability of smart contracts, mitigate security risks, and support the rapid pace of innovation in the open-source blockchain ecosystem, as shown in Figure 6.

The document lists several common tools for detecting weaknesses in smart contracts. Static analysis tools, such as Mythril and Slither, automatically scan for vulnerabilities. Formal verification tools mathematically prove the correctness of contracts. Testing frameworks, such as Truffle or Hardhat, simulate transaction execution. Runtime monitoring tools help detect unusual on-chain behavior after deployment. Auditors choose tools depending on a contract's complexity and security requirements. The combination of automated and manual analyses yielded the best results.

Example Security Audit Process Using Slither

Slither is a powerful utility for Ethereum smart contract audits because of its versatility and comprehensive reporting. It performs static code analysis without executing the contract and quickly identifies weak points, such as uninitialized variables and unsafe function usage [8]. Users can integrate Slither within the continuous integration pipeline so that every code update triggers a new audit, ensuring that vulnerabilities are caught early. The tool generates interactive and detailed reports indicating severity and suggested remediation for each issue. Slither also supports defining custom vulnerability checks tailored to a project's specific requirements, as shown in Table 2.

This flexibility is essential as smart contracts become more sophisticated and interact with diverse on-chain components. By using Slither, developers can encourage best practices and safeguard smart contracts against frequently recognized threats.

Ethereum's Role in Open Source Blockchain Development

Ethereum pioneered open collaboration in blockchain by introducing smart contracts and fostering a robust global developer community that contributes to and reviews its codebase. Its open-source approach has accelerated decentralized innovation and encouraged new standards, such as Ethereum Improvement Proposals (EIPs). Noteworthy projects such as MakerDAO, Uniswap, and CryptoKitties thrive on Ethereum, attesting to the vibrancy and adaptability of the ecosystem [9].

Ethereum's open infrastructure is inclusive, welcoming developers worldwide to propose improvements and share solutions, leading to resilient platforms and diverse applications. The move to Ethereum 2.0, transitioning to proof-of-stake with sharding for scalability and cost efficiency, demonstrates continuous collaborative progress. This philosophy has set standards for transparency and democratized technological development. This effect radiates across other blockchain projects, fostering interoperability and decentralized governance and broadening the adoption of trustless systems.

Challenges and Future Directions

Open-source blockchain development faces persistent challenges such as scalability, cross-chain compatibility, governance, and evolving security threats. Scalability is being addressed by technologies such as sharding and layer 2 solutions, which significantly boost transaction throughput and reduce congestion. However, security remains a balancing act; solutions must scale without weakening the defenses, requiring ongoing research and innovation in vulnerability detection and more sophisticated testing tools [10]. Educational outreach and broader adoption are crucial, as understanding the potential and limitations of blockchain technology empowers more effective and secure implementation. The intersection of blockchain with new technologies, such as the Internet of Things (IoT), suggests even richer possibilities, such as secure, real-time data exchanges and collaborative automation. The collaborative and open-source approach ensures that as one challenge is met, community-driven solutions can address the next challenge. The continual evolution of blockchain technology, guided by inclusive and open development, will drive a decentralized future across industries.

CONCLUSION

The study concludes that smart contract security audits are essential in blockchain-based ecosystems. As decentralized systems grow, security risks increase proportionally to the growth. Auditing ensures trust, safety, and integrity in smart contract operations. The combination of automated tools and expert human analysis results in the strongest protection against false negatives. We need to keep an eye on things all the time to ensure that contracts are safe in the run. Security is extremely important. We must consider it at every step, from when we first start working on something to when we put it out there, and even when we update it later. The document states that it is much better to take steps to prevent security problems before they occur than to try to fix them after they have already occurred. We have to make security a priority when we are developing contracts, when we are deploying contracts, and even when we are making updates to contracts.

Blockchain Hash Equation

$$H = \text{SHA-256}(\text{Block Header} + \text{Nonce} + \text{Merkle Root})$$

Consensus Probability Equation

$$P(\text{win}) = \text{Miner\_Hashpower} / \text{Total\_Network\_Hashpower}$$

Algorithm 1: Proof of Work Mining Process

1. Input: Block Header, Target Value
2. Initialize nonce = 0
3. Repeat:
   a. Compute Hash = SHA-256(Header || nonce)
   b. If Hash < Target:
      Accept block and broadcast
   Else:
      nonce = nonce + 1
      4. End Repeat

REFERENCES

1. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [white paper]. 2008. Available from: https://bitcoin.org/bitcoin.pdf
2. Buterin V. Ethereum: A next-generation smart contract and decentralized application platform [white paper]. 2014. Available from: https://ethereum.org/whitepaper/
3. Crosby M, Nachiappan, Pattanayak P, Verma S, Kalyanaraman V. Blockchain technology: Beyond Bitcoin. Appl Innov Rev. 2016 Jun;(2):6–19.
4. Luu L, Chu DH, Olickel H, Saxena P, Hobor A. Making Smart Contracts Smarter. CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 2016. p. 254–269. doi:10.1145/2976749.2978309.
5. Atzei N, Bartoletti M, Cimoli T. A survey of attacks on Ethereum smart contracts (SoK). In: Principles of Security and Trust; 2017. p. 164–186. doi:10.1007/978-3-662-54455-6_8.
6. Yaish A, Stern G, Zohar A. Uncle Maker: (Time)Stamping Out the Competition in Ethereum. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark. 2023. p. 1614–1628. doi:10.1145/3576915.3616674.
7. Gamage HTM, Weerasinghe HD, Dias NGJ. A survey on blockchain technology concepts, applications, and issues. SN Comput Sci. 2020;1(2):114. doi:10.1007/s42979-020-00123-0.
8. Wang X, He J, Xie Z, Zhao G, Cheung SC. ContractGuard: Defend Ethereum Smart Contracts with Embedded Intrusion Detection. IEEE Trans Serv Comput. 2020;13(2):314–328. doi:10.1109/TSC.2019.2949561.
9. 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. 2018. p. 67–82. doi:10.1145/3243734.3243780.
10. Sidhu J. Syscoin: A peer-to-peer electronic cash system with blockchain-based services for e-business. ICCCN: 2017 26th International Conference on Computer Communication and Networks, Vancouver, Canada, 2017. p. 1–6. doi:10.1109/ICCCN.2017.8038518.

