International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 1, Jan-Feb 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

MetaChain: Building a Web 3.0 Ecosystem with Blockchain and Wallet Interoperability in Banking Sector

Girish.M
Information Science and Engineering
CMR Institute of Technology,
Bengaluru, India
gim22ise@cmrit.ac.in

Harini.R
Information Science and Engineering
CMR Institute of Technology,
Bengaluru, India
har22ise@cmrit.ac.in

Gana Priya.V
Information Science and Engineering
CMR Institute of Technology,
Bengaluru, India
gav22ise@cmrit.ac.in

Shilpa Mangesh Pande
Information Science and Engineering
CMR Institute of Technology,
Bengaluru, India
shilpa.p@cmrit.ac.in

***Abstract*—Web3.0 has introduced a new way of building internet applications that operate without a central authority. Unlike traditional web systems that depend on centralized servers and databases, decentralized applications (DApps) provide users with greater privacy, transparency, and control over their data. This project presents a Web3.0-based decentralized application that allows users to send and receive cryptocurrency transactions securely using the Ethereum blockchain. The application combines React.js for the user interface, Solidity for the smart contract logic, and MetaMask for blockchain wallet integration. Each transaction is recorded on the blockchain, ensuring data integrity and immutability. By using smart contracts, the system eliminates the need for intermediaries, reduces security risks, and promotes trust between users. This paper explains the design, architecture, and implementation of the DApp, highlighting how blockchain technology can enhance transparency and security in web-based applications.**

***Index Terms*—Web3.0, Blockchain, Smart Contracts, Ethereum, MetaMask, Decentralized Application**

I. INTRODUCTION

The evolution of the internet has progressed from static Web1.0 to interactive Web2.0 and now to decentralized Web3.0 technologies. Web3.0 introduces decentralization, security, and transparency as its fundamental principles by leveraging blockchain technology. Unlike traditional centralized systems that depend on intermediaries for managing and validating transactions, Web3.0 allows peer-to-peer interaction, giving users full control over their data and assets.

The proposed system, *Decentralized Web3.0 Application for Secure Blockchain Transactions*, focuses on enabling secure, transparent, and immutable financial transactions using the Ethereum blockchain. It integrates smart contracts written in Solidity and deployed on the Ethereum test network to manage all transactional activities. The frontend of the system is designed using React.js and Tailwind CSS, ensuring an interactive and responsive interface. User authentication and wallet connectivity are achieved through MetaMask integration, allowing seamless communication between the user interface and blockchain network.

The main objectives of the proposed system are:

- To develop a decentralized application that ensures secure peer-to-peer cryptocurrency transactions.
- To utilize smart contracts for automating and validating blockchain operations.
- To integrate MetaMask for safe wallet connection and transaction signing.
- To maintain transparency, traceability, and immutability in all recorded data.

The paper is structured as follows: Section II presents related work on blockchain-based decentralized systems. Section III explains the methodology and architecture of the proposed DApp.

II. RELATED WORK

Several studies have examined blockchain systems, smart contracts, and Web3.0 ecosystems from software, security, and application perspectives. Wallet-centric research has gained importance, where a decentralized multi-platform wallet using blockchain and IPFS is proposed in [1] to address privacy exposure, third-party dependence, and key recovery issues in decentralized applications. At a broader level, the evolution of Web3.0 is systematically reviewed in [2], identifying core ecosystem components such as decentralized identity, NFTs, and governance, while highlighting challenges related to scalability and interoperability. From a software engineering viewpoint, continuous deployment for blockchain applications is explored in [3], introducing a DevOps-oriented “1+5 views” architecture that supports iterative updates without redeploying

core smart contracts. Complementary work in [4] focuses on improving smart contract reliability by introducing transaction primitives that constrain execution behavior and reduce nondeterminism. Smart contract design and management challenges are further discussed in [5] and [6], emphasizing issues of maintainability, requirement alignment, and contract evolution. Security-focused studies investigate blockchain-based authentication, with [7] proposing blockchain-enabled two-factor authentication, while cross-domain authentication in dynamic environments is addressed through UAV-assisted vehicular networks in [8]. Interoperability among heterogeneous blockchains is examined in [9], which proposes decentralized cross-chain interaction protocols using smart contracts and HTLCs. Application-driven research demonstrates blockchain adoption in finance and industry, including construction payment automation via BIM-integrated smart contracts in [10], financial market impacts of cryptocurrencies and DeFi in [11], and trust-centric financial automation beyond cryptocurrency use cases in [12]. From a development perspective, Solidity micro-patterns are analyzed in [13], offering reusable constructs to improve contract robustness, while blockchain–IoT integration challenges are discussed in [14]. The integration of blockchain with big data analytics for supply chain value creation is explored in [15], and transaction-level security enhancements using smart contract–controlled micro-segmentation are examined in [16]. Overall, these works demonstrate substantial progress while revealing gaps in unified transaction flow modeling and end-to-end Web3 application integration.

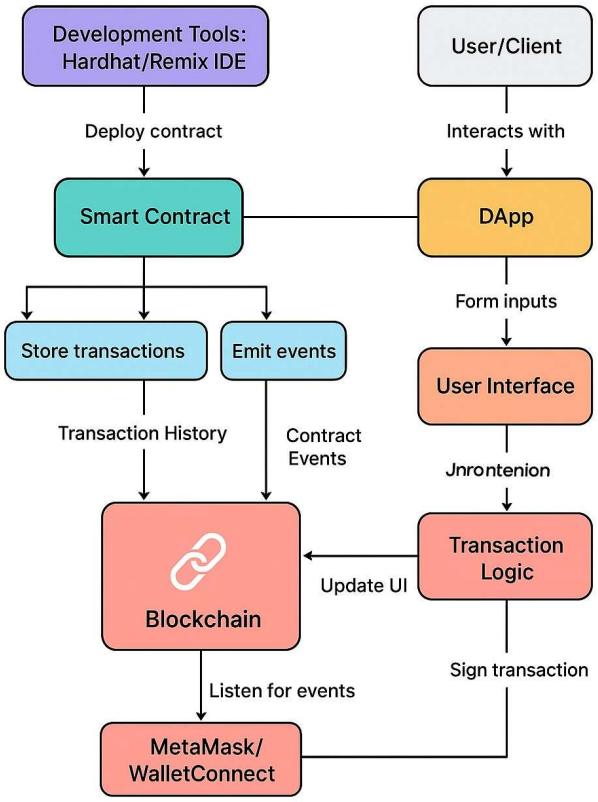

Fig. 1. Flow Diagram of Proposed Methodology

Overall, prior research highlights advancements in deployment automation, interoperability, authentication, smart contract engineering, financial decentralization, identity protection, and analytics-driven transparency. These studies form a strong foundation for modern Web 3.0 platforms, including the system developed in this work, which integrates smart contracts, decentralized interactions, and user-controlled authentication mechanisms.

III. METHODOLOGY

The proposed system implements a decentralized Web3.0 application for secure blockchain transactions. The platform allows users to initiate, validate, and store transactions without relying on centralized intermediaries. The system makes use of smart contracts, blockchain nodes, and Web3.0 protocols to ensure data security, immutability, and transparency. All transaction records are stored on the blockchain, and cryptographic mechanisms ensure privacy and integrity.

A. Transaction Processing System

To process a transaction, the user first connects to the decentralized network through a Web3-enabled interface. Transaction requests are signed using the user's private key, which is then broadcast to the network. Each transaction is validated by network nodes using a consensus mechanism before being appended to the blockchain. The system iteratively confirms transactions until they are finalized and stored on-chain.

B. Smart Contract Execution

Smart contracts automate transaction validation and enforcement of business rules. A transaction is executed when the predefined conditions in the smart contract are met. For instance, fund transfers, access permissions, or multi-party approvals are processed automatically. The contract ensures that all operations are immutable and verifiable by any participant on the network.

1) Transaction Verification: The verification module checks digital signatures, transaction amounts, and sender/receiver identities. Invalid or tampered transactions are rejected to prevent fraudulent activities.

2) Consensus Validation: The network applies the consensus algorithm (PoS, PBFT, or other mechanisms) to validate transactions. Only when the majority of nodes approve the transaction, it is committed to the blockchain ledger.

C. System Description

The system leverages several tools, libraries, and frameworks for its implementation:

(i) **Web3.js/Ethers.js** — JavaScript libraries used to connect the frontend dApp to the blockchain. They allow interaction with smart contracts, retrieval of on-chain data, and transaction signing.
(ii) **Solidity** — A high-level programming language used to write smart contracts deployed on Ethereum or compatible blockchain networks. It enables self-executing transaction rules.
(iii) **Node.js** — Used to run backend services for the dApp, including handling API calls, user authentication, and interaction with blockchain nodes.
(iv) **IPFS (InterPlanetary File System)** — A decentralized file storage protocol used to store sensitive data off-chain while ensuring immutability and integrity. Only cryptographic hashes are stored on the blockchain.
(v) **Ganache/Hardhat** — Blockchain development tools used for local testing and simulation of smart contracts before deployment to a live network.

D. Algorithmic Implementation

Transactions are executed, validated, and stored securely on the blockchain using the following procedures.

i) Transaction Recording Algorithm

Input: User transaction request (sender, receiver, amount)
Output: Validated transaction appended to blockchain

Step No.	Step Info
1	Fetch user wallet and transaction details from the decentralized application (dApp).
2	Sign the transaction using the user's private key to ensure authenticity.
3	Broadcast the signed transaction to the blockchain network.
4	Validate the transaction through the consensus mechanism (e.g., PoW or PoS).
5	Append the validated transaction to the blockchain ledger.
6	Update the user's transaction history and notify participants.

ii) Smart Contract Execution Algorithm

Input: Transaction request triggering a smart contract
Output: Executed contract and updated blockchain state

Step No.	Step Info
1	Fetch the relevant smart contract from the blockchain.
2	Verify predefined conditions in the contract.
3	Execute smart contract actions automatically once conditions are met.
4	Validate results with network nodes.
5	Store the execution outcome and transaction record on-chain.
6	Update dApp interface to reflect the executed state.

iii) MetaMask Wallet Connection Algorithm

Input: User interaction to connect MetaMask
Output: Active wallet address returned

Step No.	Step Info
1	Check if MetaMask is installed in the browser.
2	If not installed, prompt the user to install MetaMask.
3	Request the list of accounts from MetaMask.
4	Return the active wallet address to the dApp.

iv) Hardhat Smart Contract Deployment Algorithm

Input: Smart contract bytecode and ABI
Output: Deployed contract address on blockchain

Step No.	Step Info
1	Load the smart contract bytecode and ABI into the deployment script.
2	Estimate the gas required for contract deployment.
3	Broadcast the deployment transaction to the blockchain network.
4	Wait for the transaction receipt confirming deployment.
5	Store the deployed contract address for further interactions.

v) On-Chain Event Listener Algorithm

Input: Smart contract events
Output: Updated dApp UI reflecting events

Step No.	Step Info
1	Subscribe to the relevant smart contract event, e.g., `TransferEvent`.
2	For each detected event, update the dApp user interface accordingly.
3	Maintain a record of event history in the UI for user reference.

E. Security and Privacy Measures

To ensure secure and private transactions, the system uses:

- **Public-Private Key Encryption** — For authentication and signing transactions.
- **Digital Signatures** — To verify sender authenticity.
- **Hashing** — For transaction integrity and immutability.
- **Decentralized Storage** — Sensitive data stored on IPFS to prevent central point of failure.

F. Blockchain Transaction Flow

Figure 2 depicts the blockchain transaction flow of the proposed Web3.0 application. A transaction is initiated by the sender through the DApp and verified by the smart contract based on predefined rules. Once validated, the transaction is executed and permanently recorded on the blockchain, ensuring transparency, security, and immutability.

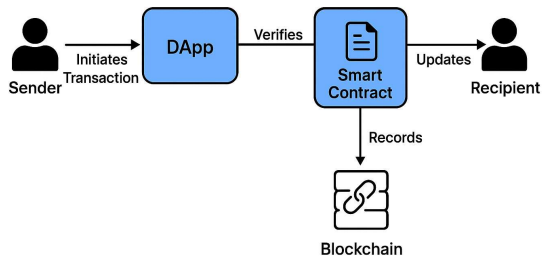

Fig. 2. Blockchain Transaction Flow

G. Dataflow Description

Figure 3 shows the overall dataflow of the proposed Web3.0 application, from transaction initiation by users to validation, smart contract execution, and final storage on the blockchain.

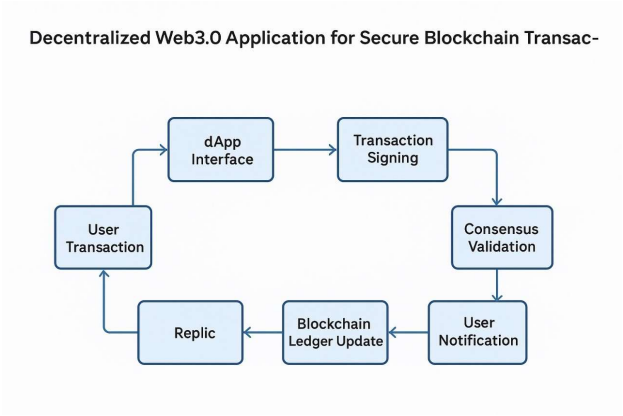

Fig. 3. Dataflow Diagram

H. Gas Cost Estimation Model

The gas consumption of a smart contract function f is estimated as:

$$\mathrm{GCE}_f = \sum_{o \in O_f} g(o) + g_{\text{storage}} \cdot S_f + g_{\text{tx}}, \tag{1}$$

$$\widehat{\mathrm{Gas}}_f = \mathrm{GCE}_f \cdot (1 + \delta). \tag{2}$$

Where:

- O_f : Set of EVM opcodes executed by function f
- $g(o)$: Gas cost of opcode o
- g_{storage} : Gas consumed per storage write/update
- S_f : Number of storage operations performed by function f
- g_{tx} : Base transaction cost (typically 21,000 gas)
- δ : Safety factor for unpredictable gas variations

This model is used to estimate runtime gas cost, prevent transaction failure, and optimize smart contract design.

I. Smart Contract Deployment Cost

The total deployment cost of a smart contract is:

$$DC = \widehat{\mathrm{Gas}}_{\text{deploy}} \cdot (\text{BaseFee} + \text{Tip}). \tag{3}$$

Where:

- $\widehat{\mathrm{Gas}}_{\text{deploy}}$: Estimated deployment gas requirement
- BaseFee : Mandatory network fee (EIP-1559)
- Tip : Priority fee paid to validators

This is essential for estimating real-world deployment cost in ETH or USD.

J. Blockchain Storage Cost Model

Smart contract storage cost is defined as:

$$C_{storage} = C_{slot} \times N_{fields}. \tag{4}$$

Where:

- C_{slot} : Gas cost per storage slot (20,000 gas for new slot)
- N_{fields} : Number of state variables stored

This model is used because storage is the costliest on-chain operation and directly influences user transaction fees.

K. Transaction Validation Model

The validity of a blockchain transaction is given by:

$$T_{valid} = \begin{cases} 1, & \text{if } (Bal_{sender} \geq A_{tx}) \wedge (Addr_{recv} \neq \emptyset), \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Account State Update Rules:

$$Bal^{new}_{sender} = Bal^{old}_{sender} - A_{tx} - G_{cost}, \tag{6}$$

$$Bal^{new}_{recv} = Bal^{old}_{recv} + A_{tx}, \tag{7}$$

$$G_{cost} = G_{used} \times G_{price}, \tag{8}$$

$$Nonce_{new} = Nonce_{old} + 1. \tag{9}$$

These equations define how wallet balances and nonces change after each transaction, which is critical for transaction integrity.

L. Smart Contract Execution Model

Smart contract execution and finalization can be described as:

$$SC_{exec} = f(\text{Inputs}, State_{old}) \rightarrow State_{new}, \tag{10}$$

$$C_{final} = \begin{cases} 1, & \text{if } B_{conf} \geq N_{req}, \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

$$TPS = \frac{N_{tx}}{T_{block}}. \tag{12}$$

This model explains how contract logic updates blockchain state and how throughput (TPS) is measured.

IV. RESULTS

The work involves developing a decentralized Web3.0 application that enables secure, transparent, and tamper-proof blockchain transactions. The system eliminates the need for intermediaries by utilizing smart contracts deployed on the Ethereum blockchain. Each transaction is validated through a distributed consensus mechanism, ensuring data integrity and immutability.

The application provides a user-friendly front-end interface that connects to the blockchain using MetaMask and Web3.js. Users can securely perform transactions, view blockchain data, and verify records in real time. The backend ensures all transaction details — such as wallet address, transaction hash, timestamp, and block number — are securely stored and retrievable for auditing.

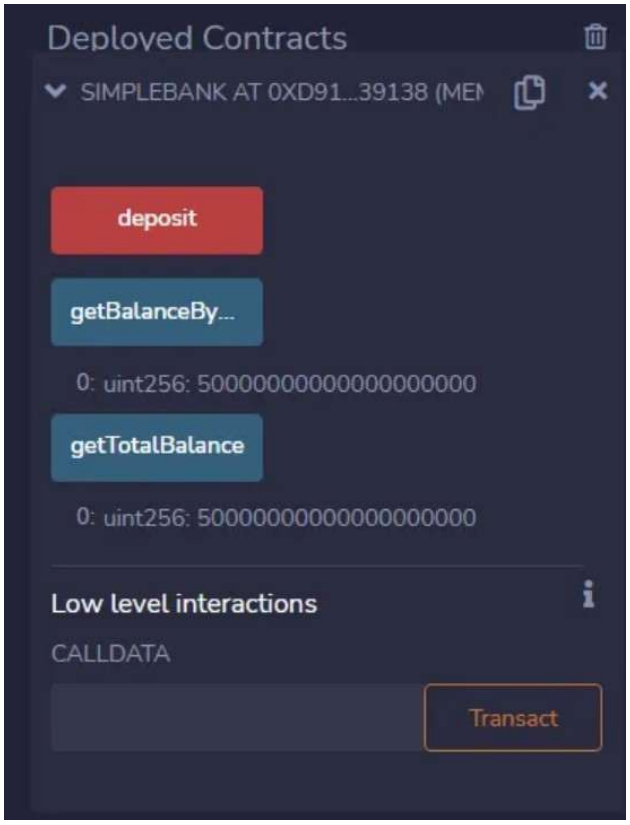

Fig. 4. Smart Contract Deployment

The interface displays the available contract functions such as deposit(), getBalanceByAddress(), and getTotalBalance(). After execution, both balance-retrieval functions return a large integer value representing the stored ether amount in Wei. This output verifies that the contract is working correctly, storing values, and returning account-specific as well as total balances on request.

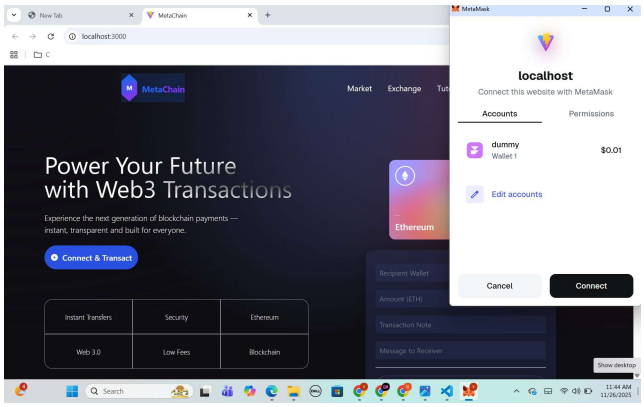

Fig. 5. Wallet Connection Request

This figure illustrates the MetaMask wallet connection prompt that appears when the MetaChain application requests access to the user's blockchain wallet. The interface displays the list of available wallet accounts along with their corresponding balances, allowing the user to select an account for interaction with the decentralized application. At this stage, the user is explicitly asked to grant permission, ensuring that wallet access is controlled and user-approved. Once the connection is accepted, the DApp gains read-only access to the user's public wallet address, which is necessary for initiating transactions, displaying account information, and interacting with deployed smart contracts. This step represents the initial handshake between the frontend application and the blockchain ecosystem, establishing a secure communication channel while preserving user ownership and privacy of cryptographic credentials.

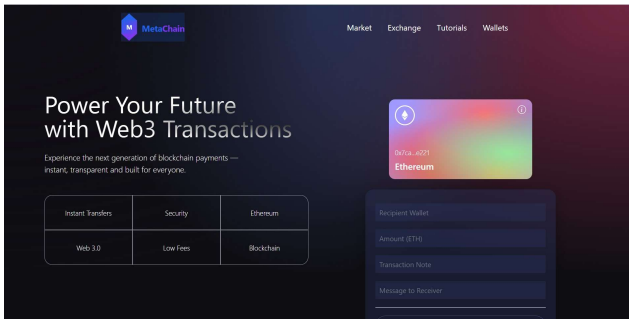

Fig. 6. Successful Wallet Connection

This shows the MetaChain homepage after the wallet has been successfully connected. The interface displays blockchain-related UI components such as the Ethereum card, the transaction form, and the navigation bar. The connection indicator is no longer visible, confirming that the DApp is fully linked to MetaMask. This result highlights the responsive UI transition that occurs once authentication is complete.

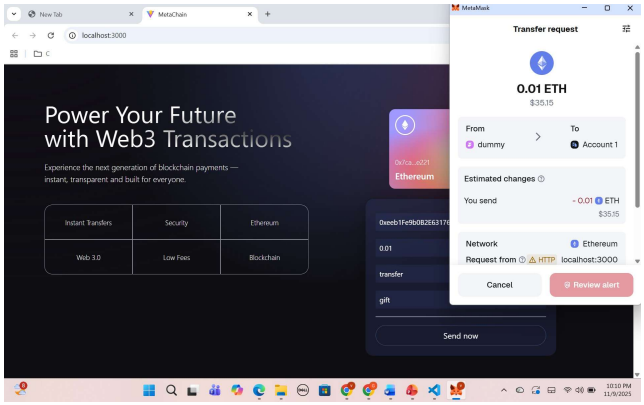

Fig. 7. Transaction Confirmation Interface

This presents the main landing page of the MetaChain application before any wallet interaction. It shows the platform's design elements, including transaction options, feature highlights (Web 3.0, low fees, blockchain), and the dedicated form for initiating payments. This serves as the baseline UI state, demonstrating the user-friendly layout developed for blockchain-based transfers.

As shown in Figures 2–7, the developed application successfully handles decentralized transactions using smart contracts. Each transaction is validated on-chain, and its hash is permanently recorded on the blockchain, ensuring transparency and immutability.

V. CONCLUSION

The proposed Decentralized Web3.0 Application for Secure Blockchain Transactions effectively utilizes blockchain technology to enhance data security, transparency, and user control. By integrating smart contracts and decentralized authentication, the system eliminates third-party dependence and minimizes transaction fraud.

The use of Web3.0 principles, such as decentralization and distributed consensus, makes the system more resilient to single-point failures. This project demonstrates a scalable and secure approach to modern blockchain-based applications.

Future work can include integration with multi-chain networks, NFT-based access control, and layer-2 scaling solutions to improve transaction speed and reduce costs.

REFERENCES

[1] C. Daude´n-Esmel, J. Castella`-Roca, A. Viejo, and I. Miguel-Rodr´iguez, "Multi-platform wallet for privacy protection and key recovery in decentralized applications," *Blockchain: Research and Applications*, vol. 6, no. 1, p. 100243, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2096720924000563

[2] C. Guan, D. Ding, J. Guo, and Y. Teng, "An ecosystem approach to web3.0: a systematic review and research agenda," *Journal of Electronic Business Digital Economics*, vol. 2, no. 1, pp. 139–156, 07 2023. [Online]. Available: https://doi.org/10.1108/JEBDE-10-2022-0039

[3] T. Go´rski, "Towards continuous deployment for blockchain," *Applied Sciences*, vol. 11, no. 24, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/24/11745

[4] S. Mansouri, H. Mohammed, N. Korchiev, and K. Anyanwu, "Taming smart contracts with blockchain transaction primitives: A possibility?" in *2024 IEEE International Conference on Blockchain (Blockchain)*, 2024, pp. 575–582.

[5] V. N. Huynh Anh, "An organizational modeling for developing smart contracts on blockchain-based supply chain finance systems," *Procedia Computer Science*, vol. 239, pp. 3–10, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050924013759

[6] H. Taherdoost, "Smart contracts in blockchain technology: A critical review," *Information*, vol. 14, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/2/117

[7] C. McCabe, A. I. C. Mohideen, and R. Singh, "A blockchain-based authentication mechanism for enhanced security," *Sensors*, vol. 24, no. 17, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/17/5830

[8] W. Wang, S. Zhang, G. Liu, and Y. Zhao, "A blockchain-based cross-domain authentication scheme for unmanned aerial vehicle-assisted vehicular networks," *World Electric Vehicle Journal*, vol. 16, no. 4, 2025. [Online]. Available: https://www.mdpi.com/2032-6653/16/4/199

[9] L. Cheng, Z. Lv, O. Alfarraj, A. Tolba, X. Yu, and Y. Ren, "Secure cross-chain interaction solution in multi-blockchain environment," *Heliyon*, vol. 10, no. 7, p. e28861, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844024048928

[10] H. Elsharkawi, E. Elbeltagi, M. S. Eid, W. Alattyih, and H. Wefki, "Construction payment automation through scan-to-bim and blockchain-enabled smart contract," *Buildings*, vol. 15, no. 2, 2025. [Online]. Available: https://www.mdpi.com/2075-5309/15/2/213

[11] U. Kayani and F. Hasan, "Unveiling cryptocurrency impact on financial markets and traditional banking systems: Lessons for sustainable blockchain and interdisciplinary collaborations," *Journal of Risk and Financial Management*, vol. 17, no. 2, 2024. [Online]. Available: https://www.mdpi.com/1911-8074/17/2/58

[12] H. Chen, N. Wei, L. Wang, W. Fawzy Mohamed Mobarak, M. Ali Al-bahar, and Z. A. Shaikh, "The role of blockchain in finance beyond cryptocurrency: Trust, data management, and automation," *IEEE Access*, vol. 12, pp. 64 861–64 885, 2024.

[13] L. Ruschioni, R. Shuttleworth, R. Neykova, B. Re, and G. Destefanis, "Micro-patterns in solidity code," 2025. [Online]. Available: https://arxiv.org/abs/2505.01282

[14] A. Rashid and M. J. Siddique, "Smart contracts integration between blockchain and internet of things: Opportunities and challenges," in *2019 2nd International Conference on Advancements in Computational Sciences (ICACS)*, 2019, pp. 1–9.

[15] A. Jabbar, P. Akhtar, and S. I. Ali, "The interplay between blockchain and big data analytics for enhancing supply chain value creation in micro, small, and medium enterprises," *Annals of Operations Research*, vol. 350, no. 2, pp. 649–671, 2025.

[16] W. A. Jebbar and M. Al-Zubaidie, "Transaction-based blockchain systems security improvement employing micro-segmentation controlled by smart contracts and detection of saddle goatfish," *SN Computer Science*, vol. 5, no. 7, p. 898, 2024.

