ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

Secure Passwordless Authentication Using Blockchain: A Cryptographic Challenge-Response Approach

Kadambari Marne[1]; Parag Jambulkar[2]; Siddhi Khandarkar[3]; Mrunal Mohite[4]

[2]Professor

[1,2,3,4]Computer Department Pune Institute of Computer Technology Pune, Maharashtra, India

Publication Date: 2025/11/06

Abstract: Traditional web authentication mechanisms depend heavily on centralized password storage, which introduces signif- icant vulnerabilities including data breaches, credential stuffing attacks, and phishing threats. This research presents a novel pass- wordless authentication framework that utilizes the non-custodial cryptographic identity capabilities of modern Ethereum wallets combined with an immutable public key registry implemented through blockchain smart contracts. Our approach fundamen- tally transforms conventional username-password validation into a secure cryptographic challenge-response protocol. The backend server generates unique, session-specific challenge strings that users sign locally using their private keys through browser wallets. Signature verification against registered public addresses enables robust, tamper-resistant authentication without exposing private keys, storing passwords, or depending on centralized secret management systems. This architecture significantly en- hances security while maintaining user convenience and system integrity.

Keywords: *Blockchain, Passwordless Authentication, Ethereum, Cryptography, Smart Contracts, Decentralized Identity, Challenge-Response Protocol, Web3, Security.*

How to Cite: Kadambari Marne; Parag Jambulkar; Siddhi Khandarkar; Mrunal Mohite (2025). Secure Passwordless Authentication Using Blockchain: A Cryptographic Challenge-Response Approach. *International Journal of Innovative Scienceand Research Technology*,10(10), 2587-2593. https://doi.org/10.38124/ijisrt/25oct1365

I. INTRODUCTION

➢ *The Password Vulnerability Challenge*

The conventional authentication paradigm has relied on users maintaining complex secrets in the form of passwords, while organizations store derivatives of these secrets using hashing algorithms. Despite decades of refinement, this model contains fundamental security weaknesses. Even with ad- vanced hashing and salting techniques, the centralized nature of credential storage makes application servers attractive tar- gets for malicious actors. Recent high-profile data breaches have repeatedly demonstrated that sensitive information re- mains only as secure as the most vulnerable component within the server infrastructure.

Beyond storage vulnerabilities, user behavior compounds these risks. Password reuse across multiple platforms remains widespread, creating cascading security failures when creden- tials from one compromised service enable credential stuff- ing attacks on others. Additionally, social engineering tactics continue proving effective, with phishing attacks successfully bypassing even sophisticated technical defenses by targeting the human element directly.

➢ *Decentralized Identity as an Alternative*

Blockchain technology and decentralized systems provide compelling alternatives through self-sovereign identity man- agement mechanisms. Within the Ethereum ecosystem specif- ically, user identity becomes intrinsically tied to cryptographic key pairs. The public address functions as a unique, verifi- able identifier, while the corresponding private key remains exclusively under user control, secured within local wallet applications.

This work describes a comprehensive, production-ready implementation of an authentication system built upon the principle of Proof of Key Ownership. Our system seamlessly integrates web portal functionality with blockchain infrastruc- ture, utilizing smart contracts as an Immutable Public Key Registry. This architectural approach eliminates the require- ment for application servers to manage sensitive authentication credentials, effectively transferring key protection responsibil- ity entirely to users, where sophisticated wallet technology provides robust security guarantees.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

II. SYSTEM ARCHITECTURE AND COMPONENT ROLES

The proposed authentication system architecture consists of three logically separated yet interconnected layers that collec- tively manage secure authentication workflows, as illustrated in Fig. 1.

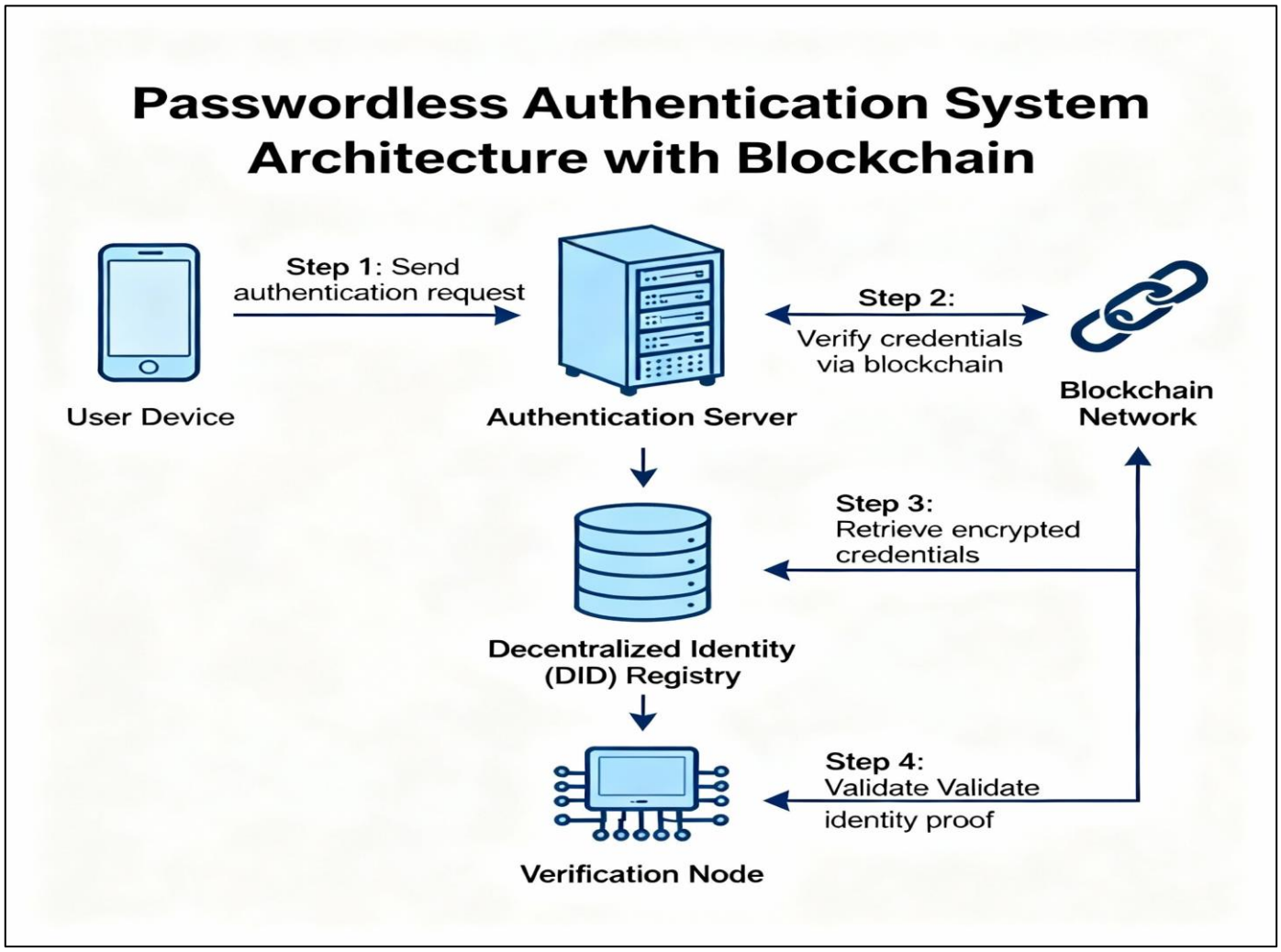

Fig 1 Passwordless Authentication System Architecture with Blockchain Showing the Interaction Between User Device, Authentication Server, Blockchain Network, DID Registry, and Verification Node.

➢ *Client Interface Layer*

The client-side component operates as an HTML and JavaScript application within the user's web browser, serving as the bridge between users and cryptographic operations.

Wallet Detection and Connection: The interface initially detects compatible non-custodial wallets such as MetaMask. Using the EIP-1193 standard, it requests access to the user's Ethereum account, establishing their public address A as their identity credential.

Challenge Management: The interface transmits A to the authentication server and awaits receipt of a session-specific challenge string C.

Cryptographic Signing: The system utilizes the wallet's eth_sign or personal_sign methods to prompt users to sign challenge C. This operation executes entirely locally using the user's private key Kpr, which never leaves the secure browser environment.

Submission Protocol: The resulting digital signature S is packaged together with the original challenge C and public address A, then submitted to the server for verification.

➢ *Authentication Server Layer*

Implemented using the Flask framework, the server-side component functions as the primary orchestrator, security gatekeeper, and blockchain interface.

Session Management: The server maintains temporary storage, such as a short-lived memory cache, to associate received public addresses A with unique, cryptographically random, time-bound challenge strings C. This mechanism proves vital for preventing replay attacks.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

Blockchain Integration: The server employs the Web3.py library to establish connections with the blockchain environ- ment. It loads the Application Binary Interface (ABI) of the deployed registry contract, enabling programmatic interaction. Verification Engine: The server hosts the critical verifica- tion endpoint with the following logic:

- Retrieval of submitted signature S and challenge C
- Application of the ecrecover cryptographic function to derive public address A′ from S and C
- Comparison of derived address A′ with expected address A
- Interaction with the Immutable Registry Layer to con- firm that A represents an authorized, registered system user

➢ *Immutable Registry Layer*

This layer represents the core blockchain innovation, imple- mented as a Solidity smart contract deployed on a controlled Ethereum network using Hardhat.

- *Purpose:*

The contract serves as a public, unmodifiable record of all authorized users within the system.

- *Functionality:*

It provides utility functions enabling system administrators to register new public addresses, along with view functions allowing the authentication server to query registration status.

- *Immutability Guarantee:*

Once deployed, both the con- tract's code and its state data, including the list of registered addresses, are secured by the underlying blockchain's consen- sus mechanism. This makes the registry impervious to single- point administrative tampering or server-side compromise.

III. IMPLEMENTATION METHODOLOGY

➢ *Development Environment*

The system was developed using a comprehensive Web3 development stack:

- *Smart Contract Development:*

Solidity version 0.8.x for contract implementation.

- *Local Blockchain Simulation:*

Hardhat framework for deploying, testing, and running the Registry Contract locally, effectively simulating a production Ethereum environment.

- *Backend Framework:*

Flask (Python) enabling rapid API development and endpoint creation.

- *Web3 Interfacing:*

Web3.py library handling RPC commu- nication between the Flask server and Hardhat node, facilitat- ing contract interaction.

- *Frontend Implementation:*

Plain HTML and JavaScript to maximize compatibility and leverage the MetaMask API directly without additional dependencies.

➢ *Contract Deployment Process*

- The deployment and integration process follows these steps: Contract Compilation: The Solidity contract undergoes compilation using Hardhat, generating both the ABI (interface description) and bytecode.
- Deployment Script Execution: A JavaScript deployment script utilizing ethers.js executes against the running Hardhat network. The script outputs the deployed contract address, which becomes essential for subsequent operations.
- Server Initialization: The Flask application initializes with two critical configuration elements: the connection URL for the Hardhat RPC endpoint, and the ABI plus deployed address of the Registry Contract. This configura- tion allows the server to instantiate a contract object us- ing web3.eth.contract and invoke functions such as Verify Registration (address A).

IV. THE CRYPTOGRAPHIC CHALLENGE-RESPONSE PROTOCOL

The authentication process fundamentally relies on Elliptic Curve Cryptography (ECC), specifically leveraging the signa- ture recovery mechanism inherent to Ethereum. The complete authentication workflow is depicted in Fig. 2.

➢ *This Cryptographic Proof Completely Substitutes Traditional Password Knowledge Verification.*

The cryptographic challenge-response flow is illustrated in Fig. 3, which demonstrates the mathematical relationship between the challenge string, private key signing operation, signature generation, and the ecrecover verification process.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

Fig 2 Secure Passwordless Authentication Flow Diagram Showing the Step-by- Step Challenge-Response Protocol with Color-Coded User Actions (Blue), Server Actions (Red), and Authorization Phase (Green).

➢ *Authentication Flow*

Table I Presents the Detailed Algorithmic Steps of the Au- Thentication Process.

➢ *Understanding the Ecrecover Function*

The ecrecover (Elliptic Curve Digital Signature Algorithm Recovery) function represents the core security mechanism. Ethereum signatures comprise three components: r, s, and v (recovery identifier). The v component enables mathematical reversal of the signing process.

Given a message (the challenge C) and signature components (r, s, v), the ecrecover function exploits ECC properties to deterministically calculate the public key used for signing. The server converts this recovered public key into the standard 40-character Ethereum address format A′. Authentication success depends entirely on the equality A′ =

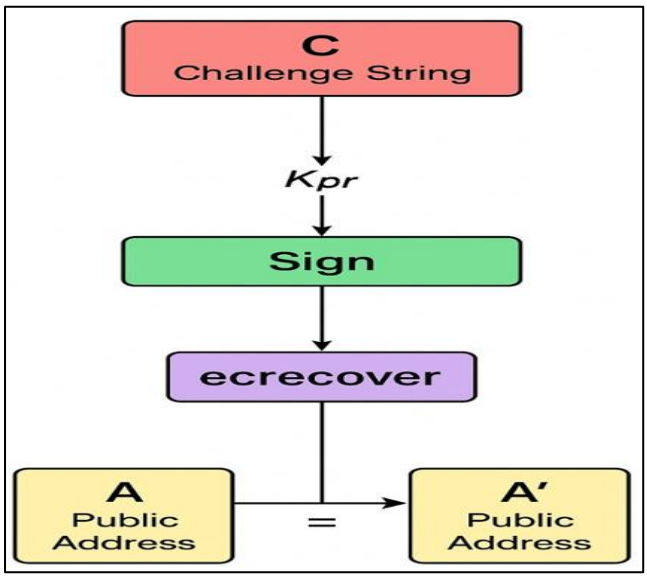

Fig 3 Cryptographic Challenge-Response Flow Diagram Showing the Pro- Cess from Challenge Generation Through Signature Creation to Address Recovery and Verification.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

V. SECURITY AND INTEGRITY ANALYSIS

This section analyzes the security guarantees provided by the proposed system against common attack vectors. Fig. ?? illustrates the security mechanisms implemented at each layer.

➢ *Replay Attack Prevention*

Replay attacks occur when adversaries capture valid authen- tication artifacts and reuse them to gain unauthorized access. Our system implements two defensive mechanisms:

- Challenge Uniqueness: Each challenge C represents a cryptographically secure random number generated fresh for every individual login attempt. A signature S remains valid exclusively for its corresponding challenge C.
- Time-Bound Validity: Challenges are stored with short expiration periods, typically 60 seconds. Verification requests arriving beyond this window result in automatic invalidation, significantly restricting exploitation opportunities.

Table 1 Detailed Authentication Flow Steps

Step	Initiator	Action	Security Context
1. Identity Decla- ration	Client	Requests wallet connection and retrieves public address A, transmitting it to server	Establishes the asserted identity A
2. Challenge Generation	Server	Generates unique 256-bit random string C and stores it temporarily, linked to A and timestamp	C functions as a nonce, preventing re- play attacks
3. Signing Oper- ation	Client Wallet	Wallet computes signature S = Sign (C, Kpr) using challenge C and user's private key Kpr. Operation is atomic and local	Kpr never transmitted; security de- pends on non-disclosure
4. Submission	Client	Sends tuple (A, C, S) to server's verification endpoint	Server receives proof of ownership S
5. Signature Re- covery	Server	Executes ecrecoveralgorithm using inputs C and S. Function returns original public address A' that created signature S	Verification: If $A' \neq A$, authentication fails immediately
6. Registry Con- firmation	Server	Queries Immutable Registry Layer: Registry. Is Registered(A)	Ensures authenticated key belongs to authorized user
7. Authorization	Server	If both recovery and registration checks pass, session token is issued and challenge C is in- validated	User successfully authenticated

➢ *Phishing and Credential Stuffing Defense*

Credential Stuffing Elimination: This attack vector be- comes entirely irrelevant since no password credentials are ever stored within the system.

- *Phishing Resistance:*

Traditional phishing attempts using fake login forms prove ineffective. Even if users inadver- tently enter their public addresses into malicious sites, those sites cannot authenticate users without possession of private key Kpr. Successful authentication requires users to approve specific signing prompts initiated by their legitimate wallet applications, which typically provide context-aware security warnings, making blind signing attacks significantly more difficult.

➢ *System Integrity Through Immutability*

Blockchain registry integration ensures that the application's most sensitive configuration component—the authorized user list—remains secured outside the application server's adminis- trative domain. Attackers compromising the Flask server might hijack active sessions but cannot:

- Add Unauthorized Users: Requires a cryptographically signed blockchain transaction with administrative privi- leges.
- Remove Legitimate Users: Similarly requires a signed blockchain transaction with appropriate authorization.

This architectural separation of concerns provides unprece- dented integrity guarantees and auditability compared to tra- ditional SQL or NoSQL user database tables.

VI. CONCLUSION

This research demonstrates a practical implementation of blockchain-based passwordless authentication that addresses several critical security challenges facing modern web ap- plications. Through our work, we've shown that combin- ing Ethereum's cryptographic capabilities with smart contract technology can create authentication systems that are both more secure and user-friendly than traditional password-based approaches.

Our implementation proves that it's entirely feasible to eliminate password storage vulnerabilities while maintaining a smooth user experience. The challenge-response protocol we've developed ensures that private keys never leave the user's device, fundamentally changing the trust model from centralized servers to individual users. This shift represents not just a technical improvement but a philosophical change in how we think about digital identity and security.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

The blockchain registry component adds an interesting layer of transparency and immutability that traditional databases simply cannot match. When user authorization data lives on- chain, it becomes auditable and tamper-evident in ways that centralized systems struggle to achieve. This doesn't mean our approach is perfect or suitable for every use case, but it does open up new possibilities for applications where auditability and decentralization matter.

Looking back at our implementation process, we found that the integration between traditional web technologies and blockchain infrastructure was smoother than initially expected. The Web3 ecosystem has matured considerably, making it re- alistic to build these kinds of hybrid systems without requiring users to understand complex blockchain concepts.

VII. FUTURE SCOPE AND RESEARCH DIRECTIONS

While our current implementation focuses specifically on authentication, we believe this work opens several promising avenues for future exploration and enhancement.

- *Extending to Data Integrity and Provenance*

One natural extension involves using the authenticated iden- tity to secure not just login sessions but all data interactions within applications. Imagine a system where every piece of in- formation submitted by users—whether it's a support ticket, a form submission, or any sensitive data—gets cryptographically signed before storage. The user's private key would create an unforgeable signature on the data hash, linking that specific user to that specific piece of information at that specific moment in time.

We're particularly interested in exploring decentralized stor- age solutions like IPFS for this purpose. Instead of storing sensitive data on centralized servers, applications could store it in a distributed manner while keeping references on-chain. The blockchain would hold minimal metadata—just the user's address, a hash of the data, and the IPFS identifier—creating a permanent, verifiable trail without bloating the blockchain with large data files.

- *Multi-Factor and Threshold Authentication*

Current wallet-based authentication relies on a single sig- nature from one private key. However, more sensitive applica- tions might benefit from threshold signatures or multi-party authentication schemes. We envision extending this system to support scenarios where multiple parties must approve an authentication attempt, or where users can split their authenti- cation capability across multiple devices for enhanced security. This becomes especially relevant for enterprise applications where different permission levels exist. A low-privilege action might require just a single signature, while administrative functions could demand signatures from multiple designated wallets. Smart contracts are well-suited to encode these kinds of complex authorization policies in an transparent, verifiable manner.

- *Cross-Chain Identity and Interoperability*

Our current implementation focuses on Ethereum, but the broader blockchain ecosystem includes numerous chains, each with unique characteristics and use cases. Future work should investigate how authenticated identities can work across different blockchain networks. Can a user authenticated on Ethereum access resources governed by smart contracts on Polygon, Arbitrum, or entirely different ecosystems like Solana?

Bridging protocols and cross-chain communication stan- dards are evolving rapidly, and authentication systems need to keep pace. We're curious about developing identity verification mechanisms that remain consistent across multiple chains while respecting the security assumptions of each individual network.

- *Privacy-Preserving Authentication*

While blockchain transparency offers benefits for auditabil- ity, it also raises privacy concerns. Every transaction and inter- action becomes permanently public. Future iterations of this system should explore zero-knowledge proof techniques that allow users to prove they're authenticated without revealing their actual public address or transaction history.

Technologies like zk-SNARKs could enable authentication where users demonstrate they possess a private key corre- sponding to an authorized address without actually exposing which specific address they control. This would maintain the security benefits of cryptographic authentication while adding meaningful privacy protections.

- *User Experience and Recovery Mechanisms*

Perhaps the biggest practical challenge with cryptographic authentication involves key management. If users lose access to their private keys, they're permanently locked out unless recovery mechanisms exist. Traditional centralized systems handle password resets through email or phone verification, but these approaches reintroduce centralization.

We need to explore robust yet user-friendly recovery mech- anisms—perhaps using social recovery where trusted contacts can collectively help restore access, or time-locked recov- ery processes that balance security with usability. Making blockchain-based authentication accessible to non-technical users remains an important unsolved challenge.

- *Performance and Scalability Considerations*

As blockchain networks evolve and gas fees fluctuate, future implementations should investigate Layer 2 solutions and alternative consensus mechanisms that reduce the cost and latency of registry operations. For applications with millions of users, even simple registry lookups can become expensive if not architected carefully.

We're interested in exploring off-chain verification mech- anisms that reduce blockchain interaction frequency while maintaining security guarantees. Perhaps users could register once on-chain but perform many authentication

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25oct1365

operations off- chain, with periodic on-chain checkpoints for security.

These future directions represent exciting opportunities to build upon the foundation we've established. Each area presents both technical challenges and practical considerations that will require careful thought and experimentation. We hope our work inspires others to explore these possibilities and push the boundaries of what's achievable with decentralized authentication systems.

REFERENCES

[1]. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, Bit- coin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press, 2016.

[2]. G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, no. 2014, pp. 1-32, 2014.

[3]. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decen- tralized Business Review, p. 21260, 2008.

[4]. M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3416-3452, 2018.

[5]. D. Boneh and V. Shoup, A Graduate Course in Applied Cryptography. Draft 0.5, 2020.

[6]. N. Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation, vol. 48, no. 177, pp. 203-209, 1987.

[7]. J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in 2012 IEEE Symposium on Security and Privacy, 2012, pp. 553-567.

[8]. M. Swan, Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc., 2015.

[9]. M. I. M. Yusop, N. H. Kamarudin, N. H. S. Suhaimi, and M. K. Hasan, "Advancing passwordless authentication: A systematic review of methods, challenges, and future directions for secure user identity," IEEE Access, vol. 13, pp. 12345-12367, 2025.

[10]. P. Khobragade and A. K. Turuk, "A gateway-assisted blockchain-based authentication scheme for internet-of-things," Journal of Network and Computer Applications, vol. 235, article 103974, 2025.

[11]. K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," IEEE Access, vol. 4, pp. 2292-2303, 2016.

[12]. M. Alharby and A. van Moorsel, "Blockchain smart contracts: Appli- cations, challenges, and future trends," Peer-to-Peer Networking and Applications, vol. 14, pp. 2901-2925, 2021.

[13]. S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 11, pp. 2266-2277, 2019.

[14]. Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3690- 3700, 2018.