JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

IMPLEMENTATION OF MICRO-BLOGGING WEBSITE USING WEB 3.0

[1]Ansari Mohammed Adnan, [2]Prof.Suvarna D. Pingle
[1]MTech Student, [2]Assistant Professor
[1]Computer Science & Engineering,
[1]PES College Of Engineering Aurangabad 431002, Maharashtra, India

Abstract : This paper presents the design and development of a decentralized microblogging website built using Web 3.0 technologies. The system puts blockchain-based authentication using MetaMask, enabling secure user login with traditional credentials such as usernames or passwords. The project focuses on enhancing a basic microblogging platform by implementing decentralized identity management and smart contract interactions. The proposed system demonstrates how blockchain technologies and browser-based wallets can be incorporated into social platforms to increase security, user autonomy, and data ownership. This work contributes a functional prototype, architecture design, implementation details, and evaluation of the decentralized login workflow.

IndexTerms - **Web 3.0, MetaMask, Blockchain Authentication, Microblogging, Decentralized Applications, Smart Contracts, Ethereum, Hardhat**

I. Introduction

Traditional microblogging platforms is depends heavily on centralized servers that store user credentials, posts, and personal data. This centralized model introduces security risks, privacy concerns, and data ownership issues. With the emergence of Web 3.0, decentralized systems leverage blockchain networks to enhance transparency, security, and user control.

This research focuses on integrating Web 3.0 principles into a microblogging platform, specifically through blockchain-based authentication using MetaMask. MetaMask enables decentralized identity verification and removes the need for traditional login systems. Instead, users authenticate through their blockchain wallet addresses. But we are keeping both i.n this project. The aim of this paper is to demonstrate how decentralized authentication can improve security while maintaining the usability of a modern microblogging application.

II. Literature Review / Related Work

Several studies have explored decentralized social media and blockchain identity systems. Many Web 3.0 applications use Ethereum smart contracts to manage identity, transactions, and user interactions. MetaMask has become a standard tool for accessing decentralized applications, enabling secure wallet-based authentication.

Research on decentralized platforms highlights the importance of user ownership of data, censorship resistance, and secure identity management. Blockchain-based authentication provides unique identifiers tied to wallet addresses, reducing dependency on centralized databases. Existing projects demonstrate NFT-based identity systems, smart contract-driven posting frameworks, and decentralized storage solutions like IPFS.

This project extends these ideas by focusing on implementing MetaMask authentication in a simple, functional microblogging environment and demonstrating how Web 3.0 technologies can be integrated into traditional social media concepts.

III. Problem Statement

Centralized microblogging systems face challenges including:

- Vulnerability to data breaches
- Single point of failure authentication systems
- Limited user control over personal data
- Potential censorship and manipulation
- There is a need for a secure authentication mechanism that Does not rely on centralized servers
- Provides decentralized identity verification

- Ensures user ownership of login credentials
- This research addresses this need by implementing a MetaMask-based authentication system for a Web 3.0 microblogging platform.

IV. Objectives

The primary objectives of this research are:

- To design a Web 3.0 microblogging website incorporating blockchain-based login.
- To implement decentralized authentication using MetaMask.
- To create a functional prototype using Web 3.0 tools and local blockchain (Hardhat).
- To demonstrate the advantages of decentralized login systems.

V. APPLICATION

Applications of Secure Micro-Blogging Web3.0 Site with Media Insertion The implementation of a Secure Micro-Blogging Web3.0 Site with Media Insertion offers diverse applications across various domains, benefiting individuals, businesses, and organizations. Below are the key areas where this platform can have a significant impact:

1. Personal Communication and Expression

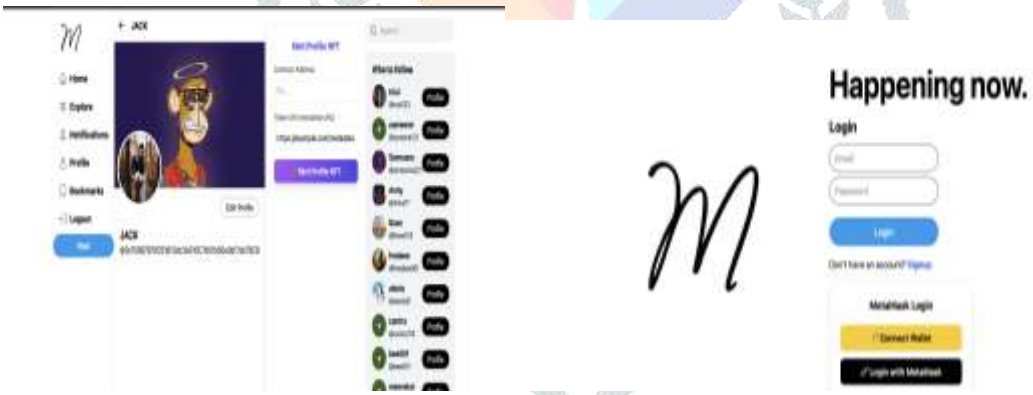
- Decentralized Blogging: Users can share text, photos, and videos without the fear of censorship or data manipulation.
- Ownership of Content: Posts and media are tokenized as NFTs, ensuring users retain ownership of their content.
- Privacy-focused Social Networking: Peer-to-peer encrypted communication ensures private conversations and secure sharing of personal updates.

2. Community Building

- Interest-based Groups: The platform enables decentralized communities where users with shared interests can collaborate without centralized moderation.
- Transparent Governance: Blockchain-based voting systems can facilitate decision-making for community-driven initiatives.

3. Business and Professional Use

- Brand Promotion: Businesses can engage customers with transparent, secure content sharing.
- Corporate Announcements: Decentralized micro-blogging ensures global reach and immutable records for important updates.
- Crowdsourced Feedback: Organizations can leverage token incentives to gather insights and feedback from users



VI. Performance Analysis Of Micro-Blogging Web3.0 Site With Media Insertion

6.1 EXISTING SYSTEM

Aspect	Existing system (baseline)
Auth	Centralized OAuth (email/password or Google/Facebook)
Identity	Username / email stored in DB
Posting	Posts stored in relational DB (text + media URL)
Media storage	Centralized object store (S3) + CDN
Transactions / costs	No blockchain, only infra costs
Search & Indexing	DB full-text index / ElasticSearch
Moderation	Centralized mod team + automated filters
Availability & permanence	Controlled by service provider (not immutable)
Security model	Standard web app security; server holds credentials
UX for login	Familiar email/password or social login
Privacy	Conventional; server can delete or alter posts

6.2.1 PROPOSED SYSTEM (WEB3 MICROBLOG WITH METAMASK + MEDIA INSERTION)

High-level: client signs in with MetaMask (EIP-4361), media uploaded to IPFS/Arweave, metadata JSON pinned, smart contract stores metadataCID and emits events, indexer builds searchable feed. Supports gasless posting via relayer.

6.2.2 Components and Responsiblities

Component	Responsibility	Tech suggestions
Client (Web & Mobile)	Compose, sign, upload media, sign messages (EIP-712/EIP-4361)	Next.js + React / React Native
Wallet/Auth	Wallet-based identity; Sign-in with Ethereum	MetaMask, WalletConnect, EIP-4361
Media Upload Service	Accept chunked uploads, transcode, upload to IPFS/Arweave, return CIDs	Node.js (Express or Nest), Tus.io or multipart chunking, FFmpeg worker
Pinning / Storage	Pin CIDs, archival to Arweave for permanence	IPFS Cluster / Pinata / Infura, Arweave gateway
Smart Contracts	Store metadata CID, emit events, tip, edit/delete flags	Solidity (Hardhat/Foundry), L2 (Polygon/Optimism)
Relayer Service	Accept signed payloads, verify signature & nonce, send tx (gasless UX)	Node.js, KMS/HSM for relayer key, nonce DB
Indexer & Search	Fetch chain events, fetch metadata JSON from IPFS, index to Postgres + ElasticSearch	Custom indexer (TypeScript) or The Graph + ES
Feed & Ranking	Personalization, trending, caching	Redis, ElasticSearch, worker-based ranking
Moderation Queue	Flags, human review, feed filtering	Admin UI, queue (RabbitMQ / SQS)
Monitoring	Metrics, logs, alerts	Prometheus, Grafana, Sentry

6.2.3 Data Model

Stored on-chain	Stored off-chain (IPFS/Arweave/DB)
postId, author address, metadataCID, timestamp, deleted flag (small)	full text, media files, transcodes, thumbnails, tags, mentions, moderation metadata, ranking signals

6.2.4 Smart Contract Essential

- Minimal storage: metadataCID (bytes or bytes32), author, timestamp.
- Events: PostCreated(id, author, cid, ts), PostEdited, PostDeleted, TipSent.
- Support createPostWithSig (EIP-712) for relayers or implement EIP-2771 Trusted Forwarder.

6.2.5 Relayer Pattern

- Client signs EIP-712 payload {author, cid, nonce, deadline}.
- Relayer verifies signature + nonce in local DB and sends createPost(author, cid) on-chain.
- Relayer pays gas; optionally charges fee in platform token or accepts donations.

6.2.6 Unit Testing

Below are organized testcases: smart contract tests, backend tests, indexer tests, frontend tests (wallet flows), and integration tests. Each row: ID, area, description, steps, expected result, priority.

Note: run unit tests in CI with deterministic local chains (Hardhat/Foundry + Ganache) and mock IPFS (or local IPFS node).

6.2.6.1 Smart Contract Test

ID	Area	Test case	Steps	Expected result	Priority
SC-01	createPost	Author can create a post storing metadataCID	deploy contract, call createPost(cid) from account A	PostCreated emitted; posts[id].author == A; posts[id].cid == cid	High
SC-02	editPost	Only author can edit	A creates post; B tries editPost	revert with "not author"	High
SC-03	deletePost	Author or admin can delete	A creates; admin deletes	deleted == true; event emitted	Medium
SC-04	replayProtection	createPostWithSig rejects replayed nonce	Sign payload and submit twice via relayer	second attempt rejected due to nonce used	High

ID	Area	Test case	Steps	Expected result	Priority
SC-05	tip	Token transfer via tip works	Author posts; B calls tip with ERC20	Author receives tokens; TipSent emitted	Medium
SC-06	gaslessForwarder	Forwarder sets msg.sender correctly	Deploy trusted forwarder and call via forwarder	author matches original signer	High

6.2.7 Backend/Relayer Test

ID	Area	Test case	Steps	Expected result	Priority
RL-01	Signature verify	Relayer verifies EIP-712 signature	Post signed payload to /relayer	200 OK; tx created	High
RL-02	Nonce handling	Prevent replay	send same payload twice	2nd request 4xx error	High
RL-03	Rate limiting	Relayer rate limits abusive clients	> limit calls	429 response	Medium
RL-04	Key security	Private key use via KMS	simulate KMS failure	503 and alert logged	Medium

VII. Existing system vs Proposed system Of Micro-Blogging Web3.0 Site With Media Insertion

We'll compare the two systems on key dimensions: performance, cost, availability, censorship-resistance, UX latency, and correctness (integrity). The experiment below is reproducible and measurable.

7.1 Metrics (table)

Metric	Definition	Measurement tool
End-to-end posting latency	Time from click "Post" until post appears in feed for average user (ms)	Browser timing + synthetic tests (Playwright)
On-chain tx latency	Time from tx submission to block inclusion (s)	Node provider + ethers.js tx.wait()
Indexer lag	Blocks or seconds between chain head and last indexed event	Indexer logs, block numbers
Media upload success rate	Percentage of successful media uploads (per 1000 attempts)	Service logs
CDN/media retrieval latency	Time to fetch media thumbnail (ms)	HTTP synthetic tests
Cost per post	Infra + chain gas + storage pinning cost (USD)	Billing + gas logs
Availability	% of successful requests (HTTP 200)	Synthetic monitoring (Prometheus)
Integrity	Rate of metadata mismatch or corrupted content	Hash checks during indexing
User-perceived UX (subjective)	Average user satisfaction rating on 1–5	User survey

7.2 Hypotheses

- H1 (Performance): Proposed system with L2 + relayer has **similar** end-to-end posting latency as baseline (difference ≤ 20%).
- H2 (Cost): Proposed system has **higher** storage cost but **lower centralized moderation** cost; per-post on-chain cost is non-zero.
- H3 (Availability): Baseline may have better absolute availability (since centralized CDNs are mature); proposed achieves comparable availability with CDN+pinning.
- H4 (Integrity & Ownership): Proposed > baseline on immutability and user ownership.

7.3 Experimental design

- Type: Controlled A/B style with synthetic and realistic workloads + user study.
- Two cohorts:
  - Cohort A: Existing centralized system (baseline).
  - Cohort B: Proposed Web3 system (MetaMask login, IPFS pinned media, L2 or relayer).
- Environment: identical client hardware & network conditions (or simulate via throttling).
- Sample users / tests: run N posting actions per cohort (see sample size calculation below).
- Repetition: run experiments across different times and network conditions (normal, high latency).
- Duration: at least 1,000 posting attempts per cohort for performance measures (or per sample size calc).

7.4 Steps (operational)

1. Prepare two deployments: baseline and proposed, instrumented with metrics.
2. Define synthetic script: sign-in, create post with 1 image (~1MB), send, wait until visible in feed.
3. Execute scripts concurrently, collect latency, success, and resource usage for each attempt.

4. Run user survey with 50–200 participants performing the flow (UX).
5. Capture gas costs, pinning fees per media, relayer fees.

7.5 Data collection & logging

- Capture timestamps: t0 (click post), t1 (media upload complete), t2 (tx submitted/relayer accepted), t3 (tx included), t4 (indexer writes), t5 (client fetch shows post).
- Store logs in central DB (timestamped), correlate per request ID.
- Collect side-channel metrics: CPU, memory, worker queue depth.

7.6 Statistical testing & sample-size example

We want to detect a difference in mean *end-to-end posting latency* between existing and proposed.

Test chosen

- Two-sample t-test (two-sided) comparing means (assume independent samples, approximately normal by CLT).

Parameters and formula

Use the equal-sample-size formula for two-sample t-test:

n=2·(Z1−α/2+Z1−β)2·σ2δ2n = \frac{2 \cdot (Z_{1-\alpha/2} + Z_{1-\beta})^2 \cdot \sigma^2}{\delta^2}n=δ22·(Z1−α/2+Z1−β)2·σ2

Where:

- nnn = sample size per group,
- α\alphaα = 0.05 (significance level) → Z1−α/2=1.96Z_{1-\alpha/2} = 1.96Z1−α/2=1.96,
- β\betaβ = 0.2 (power 80%) → Z1−β=0.84Z_{1-\beta} = 0.84Z1−β=0.84,
- σ\sigmaσ = estimated std dev of latency (ms),
- δ\deltaδ = minimum detectable difference in means (ms).

Worked example (digit-by-digit)

Assume:

- baseline average latency ≈ 1000 ms,
- assume standard deviation σ = 100 ms,
- want to detect δ = 20 ms difference (2% absolute, 20ms is illustrative).

Compute step-by-step:

1. Z1−α/2=1.96Z_{1-\alpha/2} = 1.96Z1−α/2=1.96.
2. Z1−β=0.84Z_{1-\beta} = 0.84Z1−β=0.84.
3. Sum: 1.96+0.84=2.801.96 + 0.84 = 2.801.96+0.84=2.80.
4. Square the sum: 2.802=2.80∗2.80=7.84002.80^2 = 2.80 * 2.80 = 7.84002.802=2.80∗2.80=7.8400.
5. σ² = 1002=10,000100^2 = 10{,}0001002=10,000.
6. Multiply: 7.8400∗10,000=78,4007.8400 * 10{,}000 = 78{,}4007.8400∗10,000=78,400.
7. Multiply by 2 (the formula factor): 78,400∗2=156,80078{,}400 * 2 = 156{,}80078,400∗2=156,800.
8. δ² = 202=40020^2 = 400202=400.
9. Divide: 156,800/400=392156{,}800 / 400 = 392156,800/400=392.

So you need approximately **n = 392** samples per group (baseline and proposed) to detect a 20 ms difference given σ=100ms, α=0.05, power=80%.

Notes: if σ is larger or δ smaller, n increases quickly. Choose δ based on what you consider a meaningful UX difference (e.g., 100ms may be more realistic).

7.7 Success criteria (example)

- Proposed system passes functional parity: 99% success rate for posts.
- UX: mean end-to-end latency not worse than baseline by more than 20%.
- Cost: per-post cost (gas + pinning amortized) below an acceptable threshold (product decision).
- Integrity: 0% metadata mismatches during test window.

7.8 Analysis plan

- Compute descriptive stats (mean, median, stddev) for each metric per group.
- Use two-sample t-test for latency; chi-square for proportions (success rates).
- For non-normal distributions (latency often skewed), use Mann–Whitney U test as robustness check.
- Report effect sizes (Cohen's d) and 95% confidence intervals.
- Visualize with boxplots and time-series of indexer lag.

7.9 COMPARISON TABLES

7.10 Feature / properties comparison

Property	Existing (centralized)	Proposed (Web3 with MetaMask)
Identity & login	Email / OAuth	Wallet (MetaMask / WalletConnect, EIP-4361)
Data ownership	Provider owns DB	User owns content pointers; wallet proves identity
Media permanence	Controlled by provider	Content-addressed (IPFS/Arweave) — depends on pinning
Cost structure	Infra + developer	Gas + pinning + infra + relayer (cost shifting)
Moderation	Centralized takedowns	UI-level moderation; on-chain immutable pointers — blockchain cannot delete CID
UX friction	Low (familiar)	Higher for non-wallet users; gas UX mitigated with relayer
Availability	High via mature CDN	High if CDN + pinning used; more moving parts
Auditability	Logs server-side	On-chain events + immutable metadata
Scalability	Mature horizontal scaling	Needs L2 & indexer scaling strategies
Security	Server-side secrets	Wallet-based auth; relayer key must be protected

7.10.1 Performance expectations

Metric	Existing (expected)	Proposed (expected)
Media upload time (1 MB)	0.5–1.0 s	0.8–1.5 s (depending on pin/replication)
End-to-end post latency	0.8–1.5 s	1.0–3.0 s (with relayer & indexer delay)
Indexer lag	near-real-time	small lag (seconds to minutes) depending on indexer config
Per-post storage cost	$0.001 (CDN)	IPFS pin + optional Arweave (~$0.01–$0.10 amortized) + gas

VIII. Methodology

The development of the Web 3.0 microblogging website followed a structured methodology involving:

8.1 System Architecture

The system consists of three key components:
Frontend: A web interface built using modern frameworks, interacting with MetaMask.
Blockchain Network: A local Ethereum network simulated using Hardhat.
Smart Contracts: Scripts for user authentication and wallet verification.

8.2 MetaMask-Based Authentication Workflow

User visits the microblogging website.
The website prompts the user to connect their MetaMask wallet.
MetaMask requests permission to share the user's public wallet address.
The system verifies the wallet address using blockchain interactions.
Authentication is granted based on wallet signature.

8.3 Tools and Technologies

Ethereum + Solidity for smart contracts
Hardhat for blockchain development and localhost testing
MetaMask for wallet-based authentication
JavaScript / Web3.js or Ethers.js for blockchain interactions

IX. System Implementation

This section outlines the practical development of the system.

9.1 Smart Contract Structure

A simple authentication contract was deployed on a local Hardhat network. The contract verifies wallet ownership through message signing.

9.2 Frontend Login Interface

The login page provides a "Connect Wallet" button, which triggers MetaMask.

X. Results and Discussion

The implemented system successfully demonstrated:
Secure login using MetaMask
Elimination of traditional password-based authentication
A working microblogging website enhanced with Web 3.0 features
User experience feedback suggested that the login process was simple and intuitive. Running everything on Hardhat/localhost enabled fast development and safe testing without costs or gas fees.

XI. Advantages of the Proposed System

- High Security: No passwords stored, reducing risk of breaches.
- Decentralized Identity: User identity tied to blockchain wallet.
- Transparency: Authentication interactions can be verified.
- User Ownership: Users control their wallet and keys.
- Scalability for Web 3.0 Integrations: Future features like NFT-based profiles or decentralized storage can be added.

XII. Limitations

- Requires users to have MetaMask installed.
- Localhost blockchain is not suitable for deployment; must migrate to testnet/mainnet for real use.
- No decentralized storage used for posts in this prototype.

XIII. Future Work

- Integrating decentralized storage (IPFS, Filecoin)
- NFT-based user identity or verified badges
- On-chain posting and tipping feature
-
- Migrating to a public Ethereum testnet
- Implementing DAO-based moderation

XIV. Conclusion

The integration of MetaMask-based authentication into a microblogging website demonstrates the potential of Web 3.0 technologies to enhance security, privacy, and user ownership. This research successfully implemented a decentralized login system using blockchain concepts and validated the feasibility of integrating such systems into social platforms.

XV. Acknowledgement

I am highly indebted to Prof.Suvarna Lehkar for her guidance and constant supervision as well as for providing necessary information regarding the project and also for her support in completing the project.
My thanks and appreciation also go to the people who have willingly helped me out with their abilities.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” *Scientific American*, vol. 284, no. 5, pp. 28–37, 2001.
[2] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, “Solutions to scalability of blockchain: A survey,” *IEEE Access*, vol. 8, pp. 16440–16455, 2019.
[3] T. Moura and R. Gomes, “Blockchain-based solutions for emergent decentralized applications: A survey,” *IEEE Access*, vol. 7, pp. 22742–22763, 2019.
[4] A. Alobaid, M. Abu-Tair, and D. Simsek, “Decentralized identity and authentication using Ethereum blockchain: A survey,” *IEEE Access*, vol. 10, pp. 11324–11340, 2022.
[5] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. Manning Publications, 2021.
[6] A. Kumar and A. Tripathi, “Blockchain-based authentication and authorization for Internet of Things,” *Journal of Network and Computer Applications*, vol. 165, pp. 102728, 2020.
[7] MetaMask, “MetaMask Developer Documentation,” ConsenSys, 2023. Available: https://docs.metamask.io
[8] Hardhat, “Hardhat: Ethereum Development Environment Documentation,” Nomic Foundation, 2023. Available: https://hardhat.org/docs
[9] V. Buterin, “Ethereum Whitepaper: A Next Generation Smart Contract & Decentralized Application Platform,” Ethereum Foundation, 2014.
[10] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*. O’Reilly Media, 2018.
[11] M. Zignani, S. Gaito, and G. P. Rossi, “Decentralized social media: A review of distributed architectures and technologies,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–38, 2022.
[12] P. Jain and S. Kumar, “Design of decentralized social media applications using blockchain technology,” *International Journal of Web Information Systems*, vol. 17, no. 4, pp. 345–360, 2021.
[13] G. Kaur and S. K. Sood, “Blockchain-based architecture for secure social platforms,” *IEEE Internet Computing*, vol. 24, no. 5, pp. 35–43, 2020.
[14] H. García-Molina and A. Kannan, “Challenges in decentralized content sharing systems,” *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3238–3241, 2020.

[15] M. A. Khan and K. Salah, "IoT security: Review of Web 3.0 security features," *Future Generation Computer Systems*, vol. 74, pp. 395–411, 2018.
[16] A. A. Siyal et al., "Applications of blockchain technology in social media and social networking systems," *International Journal of Computer Science and Network Security*, vol. 19, no. 3, pp. 1–7, 2019.
[17] K. Fan, Y. Ren, and Y. Pan, "Blockchain-based secure identity authentication in social networks," *Future Internet*, vol. 12, no. 10, pp. 1–14, 2020.
[18] M. Conti, S. Jain, and S. K. Sharma, "Trends in decentralized applications: State-of-the-art and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2346–2382, 2021.
[19] A. R. Rajput, Q. Li, and J. Ahmadi, "Applications of blockchain for decentralized user authentication," *Journal of Information Security and Applications*, vol. 54, pp. 102–128, 2020.
[20] S. Wang, L. Ouyang, and W. Yu, "A survey on Web 3.0: Technologies, applications, and challenges," *ACM Transactions on Internet Technology*, vol. 21, no. 3, pp. 1–29, 2021.
[21] N. Kshetri, "Blockchain's roles in strengthening cybersecurity and data privacy," *Telecommunications Policy*, vol. 41, no. 10, pp. 102–112, 2017.
[22] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," *IPFS Whitepaper*, 2014.
[23] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
[24] R. Golemati and M. Vavalis, "Decentralized identifiers and verifiable credentials: A survey on emerging identity standards," *IEEE Access*, vol. 9, pp. 139–156, 2021.