

Design and Implementation of a Smart-Contract-Based Architecture for a Blockchain-Enabled Digital Pension Certificate System

Dhramandra Sharma¹ and Monika Saxena²

¹Department of Computer Science, Banasthali Vidyapith, Rajasthan, India. Email: dhramandra.sharma@gmail.com

²Department of Computer Science, Banasthali Vidyapith, Rajasthan, India. Email: smonika@banasthali.in

Abstract

This paper presents the design and implementation methodology of a Blockchain-Enabled Digital Pension Certificate (DPC) system that combines an on-chain Ethereum-compatible layer with an off-chain application and storage layer. Using a Design Science Research (DSR) approach, the system separates immutable, cryptographically verifiable state from sensitive personal data: certificate documents are stored on the Inter Planetary File System (IPFS) with only their cryptographic hashes and minimal metadata anchored on an Ethereum-compatible Anvil/Foundry test network through Solidity smart contracts. This paper details the research paradigm, the proposed system architecture, the end-to-end certificate issuance and verification workflow, the tools and development environment, the implementation methodology including issuance and verification algorithms, and the mathematical model governing hashing, transaction structure, state transitions, verification and gas cost. The design is intended to provide a decentralized, tamper-proof, cost-efficient and role-access-controlled platform for issuing and verifying digital pension certificates in e-governance settings.

Keywords—Blockchain, Smart Contracts, Solidity, System Architecture, IPFS, Design Science Research, Digital Pension Certificate.

I. INTRODUCTION

The study is based on a structured and systematic approach to designing, developing, implementing and evaluating a digital pension certificate system, using a Design Science Research (DSR) methodological approach. The proposed blockchain-based solution is applicable to this research because it involves designing a secure and decentralized IT artifact suitable for real-world challenges such as pension certificate management, along with intensive testing and validation of that artifact. It is built on blockchain, smart contracts and a sandbox testing environment (Anvil/Foundry) to enhance transparency, data integrity, verification and security in the transaction process, mitigating the risk of transaction fraud, errors and typical paperwork.

The approach follows a complete lifecycle: problem identification, objective definition, design of the solution, system development, implementation and evaluation. It includes not only conceptual modelling but also architectural design, deployment of smart contracts, technology selection, system integration, decentralized storage analysis and security analysis, giving confidence that the suggested system is both theoretically sound and deployable in a real government environment. The experimental evaluation is carried out with the Anvil and Foundry testnets, where smart contracts are deployed and transactions are made while measuring gas usage and system performance under a repeatable scheme. The system combines an Ethereum blockchain and smart contracts with off-chain application servers and databases to address transparency, efficiency, storage and scalability requirements simultaneously. Cryptographic hashing techniques, digital signatures and role-based access control ensure data integrity and security, guaranteeing that data remains unchanged and that users only access their assigned roles. This is accomplished by placing only cryptographic evidence on the blockchain, while the important information of pensioners and certificate documents is stored off-chain in a secure system.

II. RESEARCH DESIGN AND METHODOLOGICAL APPROACH

The method adopted in this study is Design Science Research (DSR), which focuses on systematic design of technological products that solve real problems. DSR was particularly appropriate because this research aims not only to understand the currently prevalent problems of centralization, transparency, security and inefficiency in pension administration, but also to develop and test functional solutions. The research design consists of the

following key stages: problem identification, conceptual modelling, system architecture design, smart contract development, prototype implementation, and controlled evaluation. Each phase was planned to ensure methodological rigor, logical results and repeatability; the research goals and problem context were pre-defined, and design decisions were made so that the problem context and proposed solution align. Controlled experimentation was conducted to assess system behavior under controlled conditions, ensuring reliable, non-arbitrary outcomes.

The choice of Design Science Research was motivated by its favorable applicability to blockchain-based systems, which are characterized by the creation and testing of technological artifacts rather than the testing of hypotheses. DSR facilitated the structured design, development and evaluation of the DPC system using an engineering-paradigm research approach, allowing blockchain technology, smart contracts, access control and privacy-preserving features to be rolled out in a regulated, orderly way that addresses centralized data management, integrity, transparency and security. It also allowed for iterative design modifications and experimental testing in a controlled Anvil/Foundry environment, enabling practical testing of security, system performance, immutability and transaction efficiency. Both qualitative and quantitative methods were used to comprehensively study the proposed blockchain platform. The qualitative method supported conceptual understanding, system design and security analysis, and was used to develop the structural and architectural elements of the system, including blockchain architecture, on-chain/off-chain task distribution, smart contract logic and role-based access control, as well as assessment against confidentiality, integrity, authentication, non-repudiation and auditability requirements and potential threat scenarios such as data tampering, unauthorized issuance and identity impersonation. The operational performance and efficiency of the system were evaluated under controlled experimental conditions using a quantitative method, with key performance indicators including gas usage for smart contract deployment and execution, blockchain transaction processing time, and off-chain metrics such as database response time. This mixed-methods approach ensured that both the conceptual and practical viability of the system were taken into consideration, enhancing the credibility, robustness and validity of the research.

III. PROPOSED SYSTEM ARCHITECTURE

The recommended system is built as a blockchain architecture that makes digital pension certificates manageable, transparent and secure, aiming to answer questions of trust, scalability and regulatory compliance by combining decentralized blockchain components with traditional backend services. The on-chain layer is responsible for storing immutable records, cryptographic evidence and access-control policies through smart contracts that govern the certification process, approvals and state transitions. The off-chain layer handles user authentication, document processing, data storage and business-logic execution without compromising the confidentiality of sensitive information, alleviating the computational load on the blockchain.

The methodological workflow starts with the onboarding of beneficiaries via a decentralized application, followed by the secure submission of supporting documents. Authorized issuers check the eligibility of beneficiaries and issue certificates; to ensure authenticity and tamper-free status, the document's hash is calculated and recorded on-chain. Verifiable credentials are issued upon approval and all transactions are recorded, ensuring traceability, auditability and transparency throughout the certificate's life. The process has a high degree of structuring and a low level of human intervention, ensuring homogeneity and allowing third-party verification. A clear line of distinction is drawn between on-chain tasks — logging transactions, maintaining cryptographic hashes, access-control policies and public transaction records — and off-chain tasks — verifying user accounts, processing documents, maintaining databases and implementing business logic — with no sensitive personal information stored on-chain. This division lowers on-chain storage costs, boosts scalability, speeds up transactions and increases maintainability by allowing the application to be updated without redeploying core smart contracts.

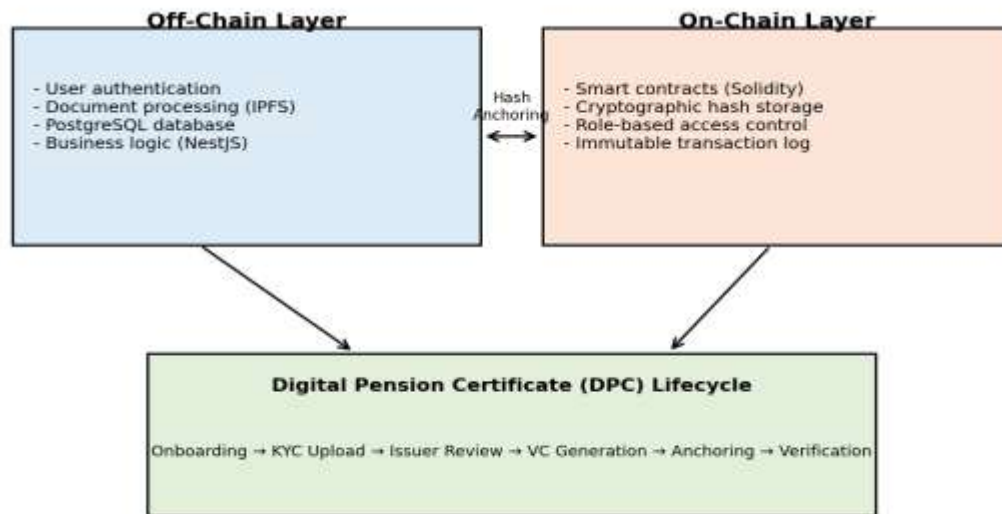


Fig. 1. On-chain / off-chain architecture separation in the proposed Digital Pension Certificate system.

The system inputs comprise pension data (pensioner names, pension ID, pension history, retirement information and eligibility records) and Aadhaar/identity data used to authenticate beneficiaries via government-issued identity proof, improving identity verification and eliminating duplication or impersonation risk. All input data is validated off-chain before further processing to guarantee accuracy, completeness and adherence to governance rules. The system outputs are a verified, digitally signed pension certificate generated upon successful verification of the beneficiary, and a blockchain hash — a cryptographic fingerprint created using SHA-256 from the certificate and its metadata and stored on the blockchain — which uniquely identifies the certificate and detects tampering upon verification.

IV. SYSTEM WORKFLOW

The end-to-end workflow proceeds through nine stages. (1) User Registration and Login: the pensioner logs in using Aadhaar/ID-based authentication or a blockchain wallet; the system verifies identity using a digital signature and role-based access control (RBAC), and assigns a role of Pensioner, Issuer or Auditor. (2) Application Submission: the pensioner uploads required documents (life proof, ID, pension details), which are accepted and forwarded for backend validation. (3) Off-Chain Storage (IPFS): all documents are stored in IPFS, and the system generates a unique Content Identifier (CID), ensuring decentralized, tamper-resistant storage. (4) Hash Generation: the system computes a SHA-256 cryptographic hash of the documents, which acts as a digital fingerprint; only the hash, not the full document, is sent to the blockchain. (5) Issuer Verification: an authorized government or bank officer reviews the application; if valid it is approved, otherwise it is rejected with a status update, and the decision is logged. (6) Smart Contract Execution: on approval, a smart contract is triggered and stores the document hash, CID, timestamp and hashed user ID as a permanent blockchain transaction. (7) Verifiable Credential (VC) Generation: the system generates a digitally signed certificate including pension details, the blockchain transaction ID and the issuer's signature as official digital pension proof. (8) Certificate Distribution: the pensioner receives the digital certificate through a dashboard, mobile wallet or PDF export. (9) Audit and Logging: every action — login, approval/rejection, blockchain transaction — is recorded to ensure transparency and accountability.

V. TOOLS, TECHNOLOGIES AND DEVELOPMENT ENVIRONMENT

To provide a secure, efficient and reproducible implementation, the proposed system uses the tools and technologies summarized in Table I. The blockchain layer is deployed on a local Ethereum-compatible test network

via Anvil, and smart contracts are written in Solidity to establish rules for issuing certificates, access control and state transitions. Smart contract compilation, deployment, testing and interaction are handled by the Forge and Cast toolchain built by Foundry, which allows contracts to be executed deterministically across repeated experimental runs, simplifying debugging, testing and rapid deployment. The backend application is built with the NestJS framework on Node.js/TypeScript, managing business logic, processing user requests and communicating with blockchain smart contracts to create and verify certificates, while the frontend is built as a decentralized application (DApp) in Next.js, connected to wallet solutions such as MetaMask SDK, WalletConnect and Wagmi for secure blockchain interaction without requiring users to understand blockchain protocols directly. Structured application data is stored in PostgreSQL, while sensitive documents are stored in a tamper-proof, content-addressable manner in IPFS via Pinata; only cryptographic hash references to these documents are kept on-chain, decreasing on-chain storage requirements. Security is preserved through SHA-256 hashing and the Elliptic Curve Digital Signature Algorithm (ECDSA), providing data integrity, non-repudiation and authentication. The complete environment, including RPC endpoints, IDE and runtime configuration, enables controlled execution, systematic testing and reproducible evaluation of the proposed blockchain-based system.

The blockchain and smart contract layer uses an Ethereum-compatible local blockchain environment built with Anvil. Smart contracts written in Solidity define certificate-issuance logic, implement role-based access control, and implement state changes during the certificate's lifecycle. The Forge and Cast toolchain from Foundry is used for compiling, deploying, testing and interacting with the smart contracts, allowing deterministic execution across repeated experimental runs without changes in contract behaviour, and simplifying debugging and rapid redeployment. Combining Anvil with Foundry allows all experiments to be conducted in a controlled environment that accurately mimics Ethereum network behaviour while retaining full control over test settings, supporting the transparency, repeatability and verifiability that academic research on blockchain operations requires.

The backend and application layer is built using the NestJS framework on Node.js and TypeScript with a modular architecture, exposing RESTful APIs and implementing the core business rules of the proposed system, coordinating communication between the frontend application, the blockchain network, the database and decentralized storage services. It handles user requests, validates application data, performs certificate issuance and verification, and manages interaction with the Ethereum smart contracts, as well as authentication, transaction processing and off-chain data management. Because the backend performs all business logic without requiring blockchain computation, on-chain processing needs are lowered, system performance is improved, and scalability and maintainability are increased.

The frontend and DApp layer is built as a decentralized application using Next.js, a React-based framework chosen for its efficiency, scalability and responsiveness. It is designed to be compatible with wallet solutions such as MetaMask SDK, WalletConnect and Wagmi, enabling users to log in securely and conduct blockchain transactions through a familiar wallet interface. The frontend abstracts the complexity of the blockchain, offering easy-to-use interfaces for certificate registration, issuance, verification and administrative operations, so that users can interact with blockchain services without needing a technical understanding of blockchain protocols or smart contracts.

Off-chain storage and data management rely on PostgreSQL for reliable relational storage of user profiles, certificate data, transaction details and administrative records, while Pinata is used to protect the Interplanetary File System (IPFS) from tampering and to provide secure, decentralized, content-addressable storage of sensitive document files. These documents are never stored on-chain; only cryptographic hash references are stored, substantially decreasing the on-chain storage requirement. Separating metadata from document files in this way increases both privacy and security, since documents can be securely confirmed through blockchain-based hash validation.

Cryptography and security tooling centres on the SHA-256 hashing technique, used to generate unique cryptographic signatures for digital pension certificates and documents so that unauthorized changes to stored documents can be detected and data integrity preserved, together with the Elliptic Curve Digital Signature Algorithm (ECDSA), which secures the digital signatures of all blockchain transactions and guarantees authentication, authorization and transaction validation. Together, hashing and digital signatures provide a comprehensive security foundation that enhances trust, transparency and reliability in a blockchain-based pension certificate management system.

The execution and development environment is a local development system set up under controlled conditions to produce reproducible and repeatable experimental results. RPC endpoints connect the backend application, the frontend interface and the local Ethereum blockchain, and development, testing, debugging and deployment are carried out in a standard integrated development environment using the Foundry toolchain. Controlled runtime configurations allow the same configuration to be replicated across every iteration of the experiments, so that activities, performance characteristics and transaction-processing efficiency can be assessed consistently, reducing external variability and increasing the external validity, replicability and credibility of the experiments performed on the proposed digital pension certificate system.

VI. SYSTEM IMPLEMENTATION: WORKFLOW STEPS AND ALGORITHMS

The implementation methodology follows seven structured steps. Step 1, User Onboarding and KYC Submission: users connect a blockchain wallet through the DApp and submit Know Your Customer (KYC) details; role-based access control (RBAC) mechanisms restrict actions such as verification, approval or certification to permitted entities only. Step 2, IPFS-Based Document Storage: each submitted document is uploaded to IPFS and assigned a unique Content ID (CID) that serves as a cryptographic reference for decentralized storage, so that any modification produces a new CID and is immediately detectable. Step 3, Issuer Approval and Rejection Workflow: authorized issuers review each submitted application against a well-documented workflow structure, and all issuer actions are recorded for complete traceability and accountability. Step 4, Verifiable Credential (VC) Generation: once approved, a standardized, cryptographically signed and self-verifiable digital credential is generated. Step 5, Blockchain Anchoring of Credential Hashes: the credential hash is anchored on the blockchain via a smart contract so that it is publicly visible and cannot be altered, allowing third-party verification without exposing personal or sensitive information. Step 6, Transaction Logging and Status Finalization: transaction logs are updated with each new blockchain entry, producing a time-stamped, non-removable history that supports compliance monitoring and regulator or auditor oversight.

Two algorithms formalize the core issuance and verification logic. Proposed Algorithm 1, Pension Certificate Issuance (Input: pensioner details, certificate PDF; Output: blockchain transaction hash): (1) collect pensioner details; (2) store personal data securely off-chain; (3) upload the certificate PDF to IPFS to generate a CID; (4) generate a SHA-256 hash of the certificate; (5) call the smart contract function `storeCertificate()` with the hash and metadata; (6) receive the blockchain transaction hash confirming storage; (7) store the transaction hash and CID in the database; (8) return a confirmation message. Proposed Algorithm 2, Pension Certificate Verification (Input: certificate PDF; Output: Valid/Invalid): (1) upload the certificate PDF for verification; (2) compute the SHA-256 hash of the uploaded certificate; (3) query the smart contract function `verifyCertificate()` with the computed hash; (4) if the hash exists on-chain and the status is valid, return "Certificate Authentic", else return "Certificate Invalid or Tampered"; (5) log the verification activity, including timestamp and status, in the off-chain database. Proposed Algorithm 3 formalizes the full system as a set of functions — `User_Login()`, `Submit_Application()`, `Verify_Application()`, `Issue_Certificate()`, `Generate_VC()`, `Verify_Certificate()` and `Audit_Log()` — each guarded by role checks and each recording its outcome for the audit trail.

Proposed Algorithm 3 formalizes the complete system as an executable procedure. Begin System: Initialize Blockchain Network; Initialize Smart Contracts; Initialize IPFS Storage; Initialize Backend Server; Initialize RBAC Security Module. Step 1 – User Authentication: Function `User_Login(wallet_address)`: verify wallet signature; assign `user_role` (Pensioner / Issuer / Auditor); End Function. Step 2 – Data Submission: Function `Submit_Application(user_data, documents)`: IF `user_role` != Pensioner THEN Return "Access Denied"; End If; Store documents OFF-CHAIN in IPFS; Generate CID for each document; Compute SHA256_Hash for all documents; Send request to Issuer Module; End Function. Step 3 – Issuer Verification: Function `Verify_Application(application_id)`: IF `user_role` != Issuer THEN Return "Unauthorized Access"; End If; Review application data; IF data is Valid THEN Approve application; Else Reject application, Log status, Exit Function; End If; End Function. Step 4 – Smart Contract Execution: Function `Issue_Certificate(Hash, CID, Metadata)`: Call `Smart_Contract.StoreCertificate(Hash, Metadata)`; Receive `Transaction_Hash`; Store (`Transaction_Hash`, `CID`, `Metadata`) in Off-Chain Database; Return

Transaction_Hash; End Function. Step 5 – Certificate Generation: Function Generate_VC(Approved_Data): Create Verifiable_Credential (VC); Sign VC using digital signature; Return VC; End Function. Step 6 – Verification Process: Function Verify_Certificate(Input_Certificate): Compute Hash_Input = SHA256(Input_Certificate); Fetch Hash_Blockchain from Smart_Contract; If Hash_Input == Hash_Blockchain THEN Return "Certificate Valid"; Else Return "Certificate Invalid or Tampered"; End If; End Function. Step 7 – Logging: Function Audit_Log(activity): STORE activity WITH timestamp in database; End Function. End System. This procedural formalization consolidates the onboarding, submission, issuer-verification, smart-contract execution, credential-generation, verification and audit-logging logic described in Steps 1-6 into a single role-guarded execution flow, in which every state-changing function first checks the caller's assigned role before proceeding, and every operation that changes system state is written to the audit log.

VII. MATHEMATICAL MODEL

The design of the proposed system is mathematically modeled as an integration of cryptographic hashing functions, blockchain transaction structure, smart-contract-based state transitions and performance evaluation metrics, ensuring the integrity of data from system start to finish. The overall system is represented as $S = \{U, D, B, C, T, F\}$, where U is the set of users (Pensioners, Issuers, Auditors), D is the set of digital documents, B is the blockchain network, C is the set of smart contracts, T is the set of transactions and F is the verification function (Eq. 1). Each document is transformed into a secure cryptographic hash using SHA-256, $H_d = \text{SHA256}(D)$ (Eq. 2), and for multiple documents the system ensures integrity through a consolidated hash, either as a summation-based representation $H_{\text{total}} = \sum \text{SHA256}(D_i)$ (Eq. 3) or, preferably, a secure concatenation-based hash $H_{\text{total}} = \text{SHA256}(D_1 \parallel D_2 \parallel \dots \parallel D_n)$ (Eq. 4) that binds all documents into a single immutable hash for tamper resistance.

Each pension certificate issuance is recorded as a blockchain transaction $T = (H_{\text{total}}, \text{CID}, t, \text{IDu})$ (Eq. 5), where CID is the IPFS content identifier, t is the transaction timestamp and IDu is the hash of the user's identity, ensuring traceability, immutability and secure linkage between on-chain and off-chain data. The blockchain state evolves through smart contract execution as $St+1 = f(St, T)$ (Eq. 6), where St is the current blockchain state, T is a new transaction and f is the deterministic state transition function executed by smart contracts, guaranteeing that every state update is deterministic, auditable and consensus-driven. The validity of a submitted certificate is verified using $F(V) = 1$ if $\text{SHA256}(V) = H_{\text{block}}$, else 0 (Eq. 7), where V is the submitted certificate and H_{block} is the hash stored on the blockchain; a return value of 1 indicates a valid certificate, while 0 indicates tampering or invalidity. Since verification involves recomputing a fixed-length hash and comparing it to a stored on-chain value, its computational complexity is $O(1)$, meaning verification time remains constant regardless of certificate size and supports near real-time authenticity checks with high scalability for large user bases.

System performance is modeled through a latency equation $L_{\text{total}} = L_{\text{auth}} + L_{\text{upload}} + L_{\text{verify}} + L_{\text{block}}$ (Eq. 8), where the terms denote authentication delay, IPFS upload time, issuer verification time and blockchain confirmation time respectively, and a gas cost model $G_{\text{total}} = G_{\text{base}} + G_{\text{storage}} + G_{\text{execution}}$ (Eq. 9), where G_{base} is the base transaction fee, G_{storage} is the cost of storing hash and metadata, and $G_{\text{execution}}$ is the smart contract execution cost. These models are used respectively to evaluate system efficiency and scalability under real-world conditions and to analyse the economic feasibility of deploying the system on blockchain platforms; both are applied in subsequent experimental evaluation of the implemented prototype.

VIII. CONCLUSION

This paper has presented the design and implementation methodology of a smart-contract-based, blockchain-enabled digital pension certificate system built on a Design Science Research approach. The proposed architecture separates on-chain cryptographic evidence from off-chain sensitive data, defines a nine-stage certificate issuance and verification workflow, specifies the tools and development environment used for a reproducible Anvil/Foundry-based implementation, and formalizes the issuance and verification logic through explicit algorithms and a mathematical model covering hashing, transaction structure, state transitions, verification complexity, latency and

gas cost. The resulting architecture provides a decentralized, tamper-proof and role-access-controlled foundation whose performance, gas efficiency and security characteristics are evaluated experimentally in subsequent work.

TABLE I. TOOLS AND TECHNOLOGIES USED IN THE PROPOSED SYSTEM

Component	Tool / Technology	Purpose
Blockchain Platform	Ethereum (Local Network – Anvil)	Local blockchain environment for deployment and testing
Smart Contract Language	Solidity	Development of blockchain smart contracts
Smart Contract Toolchain	Foundry (Forge & Cast)	Smart contract compilation, deployment, testing and interaction
Backend Framework	NestJS (Node.js/TypeScript)	RESTful APIs, business logic, and blockchain integration
Frontend Framework	Next.js	Development of the decentralized web application (DApp)
Wallet Integration	MetaMask SDK / WalletConnect	User authentication and multi-wallet connectivity
Web3 Library	Wagmi	Frontend interaction with Ethereum blockchain
Relational Database	PostgreSQL	Storage of structured application data
Decentralized Storage	IPFS (with Pinata pinning)	Secure, tamper-proof, content-addressable document storage
Cryptographic Algorithm	SHA-256	Document hashing and integrity verification
Digital Signature	Ethereum ECDSA	Authentication, authorization, and transaction signing
Communication Protocol	JSON-RPC	Communication between application and blockchain node
Development / Version Control	VS Code, Git, Node.js runtime	Source-code development, debugging, and version tracking

REFERENCES

- [1] I. Ahmed, K. Toyoda, T. Nakano, S. Kasahara, S. R. Goyal, and T. H. Tran, "A Systematic Review on Blockchain-Enabled eKYC: Leveraging SSI and DID for Secure and Efficient Identity Verification," *IEEE Internet Things J.*, vol. 12, no. 21, pp. 44381-44401, 2025.

- [2] J. Ahn, E. Yi, and M. Kim, "Blockchain Consensus Mechanisms: A Bibliometric Analysis (2014-2024) Using VOSviewer and R Bibliometrix," *Inf.*, vol. 15, no. 10, pp. 1-29, 2024.
- [3] W. S. Lee, A. John, H. C. Hsu, and P. A. Hsiung, "SPChain: A Smart and Private Blockchain-Enabled Framework for Combining GDPR-Compliant Digital Assets Management With AI Models," *IEEE Access*, vol. 10, pp. 130424-130443, 2022.
- [4] I. Radeva and I. Popchev, "Blockchain-Enabled Supply-Chain in Crop Production Framework," *Cybern. Inf. Technol.*, vol. 22, no. 1, pp. 151-170, 2022.
- [5] V. Malik et al., "Building a Secure Platform for Digital Governance Interoperability and Data Exchange Using Blockchain and Deep Learning-Based Frameworks," *IEEE Access*, vol. 11, pp. 70110-70131, 2023.
- [6] K. Samunnisa, S. Vijaya, and K. Gaddam, "Blockchain-Based Decentralized Identity Management for Secure Digital Transactions," vol. 1, no. 2, pp. 22-29, 2023.
- [7] D. Sharma and M. Saxena, "Different Cryptographic Hash Functions for Security in the Blockchain," 2023 Int. Conf. Data Sci. Netw. Secur. (ICDSNS), Tiptur, India, 2023, pp. 1-6.
- [8] D. Sharma and M. Saxena, "Blockchain-Powered Pension Framework for Transparent Disbursal," presented at the Int. Conf. Emerging Challenges and Research Frontiers in Digital Transformation (ICECRF 2026), Vikrant University, Gwalior, India, Jan. 30-31, 2026.
- [9] D. Sharma and M. Saxena, "Blockchain based digital pension system for secure communication," Manuscript under review, *Blockchain: Research and Applications*, 2026.
- [10] D. Sharma and M. Saxena, "Design of a blockchain-enabled framework for digital pension certificate scheme," *Int. J. Innov. Res. Growth*, vol. 15, no. 2, pp. 152001-152009, 2026.
- [11] D. Sharma and M. Saxena, "Enhancing Transparency and Security in Pension Fund Systems using Blockchain Technology: A Theoretical Analysis," *J. Comput. Anal. Appl. (JoCAAA)*, vol. 33, no. 7, pp. 1781-1785, 2024.