# Secure And Transparent DNS Resolution Using Blockchain Technology

Priyadharshini M[1], Sakthivelan M[2], Selvakumar S[3], Shakthi K[4]
[1]B.E., M.E. (Ph.D.), Faculty, Dept of CSE, SRM Valliammai Engineering College, Chennai, India
[2,3,4] Student, Dept. of CSE, SRM Valliammai Engineering College, Chennai, India

***Abstract*— The Domain Name System (DNS) remains vulnerable to attacks such as spoofing, cache poisoning, and unauthorized record manipulation due to its centralized and insecure design. This paper presents a blockchain-based framework for secure and transparent DNS resolution using Python and Flask technologies. Each domain-to-IP mapping is stored as an immutable block containing a hash, timestamp, and previous link, ensuring data integrity and tamper resistance. A lightweight blockchain ledger integrated with a custom DNS resolver and a web-based interface enables real-time verification and record management. Experimental results demonstrate reliable query performance with minimal latency, enhanced security, and full transparency in DNS operations, thereby improving trust and resilience across networked systems [1], [2].***

***Keywords*— Blockchain, DNS Security, Distributed Ledger, Cryptographic Hashing, Flask Framework, Python, Decentralized Network, Data Integrity.***

## I. INTRODUCTION

The Domain Name System (DNS) is a fundamental part of Internet infrastructure, enabling users to access websites using readable domain names instead of numerical IP addresses [1]. However, the traditional DNS architecture is largely centralized, making it vulnerable to security threats such as DNS spoofing, cache poisoning, and unauthorized record modification [2]. These attacks compromise the integrity and authenticity of Internet communication, allowing malicious actors to redirect users to fraudulent or harmful websites.

The proposed system, Secure and Transparent DNS Resolution using Blockchain Technology, introduces a decentralized approach to DNS management that ensures immutability, transparency, and trust. By storing domain-to-IP mappings as blocks within a blockchain, each entry is cryptographically linked to the previous one using a hash function, preventing data tampering and unauthorized changes [3].

The system is developed using Python, with a lightweight blockchain implementation, a custom DNS resolver, and a Flask-based web interface for interactive record management. This setup allows users to add, query, and verify domain records while maintaining integrity across all transactions. The blockchain ensures that any unauthorized modification to DNS data breaks the hash chain, immediately signaling inconsistency [4].

By combining blockchain principles with modern web technologies, the proposed solution enhances DNS reliability and transparency, eliminates single points of failure, and provides verifiable trust without depending on centralized authorities [5]. Experimental testing shows minimal query latency and high consistency, establishing this model as a practical step toward decentralized Internet infrastructure [6], [7].

## II. PREVALENT SYSTEMS

The traditional Domain Name System (DNS) architecture operates in a centralized and hierarchical manner, where records are maintained and controlled by authoritative servers and registries [8]. While this structure supports scalability, it introduces several weaknesses, particularly in terms of security, transparency, and single-point dependency [9]. The conventional DNS does not possess inherent mechanisms to verify data authenticity or detect unauthorized modifications, making it highly susceptible to attacks such as DNS spoofing, cache poisoning, and man-in-the-middle intrusions [10].

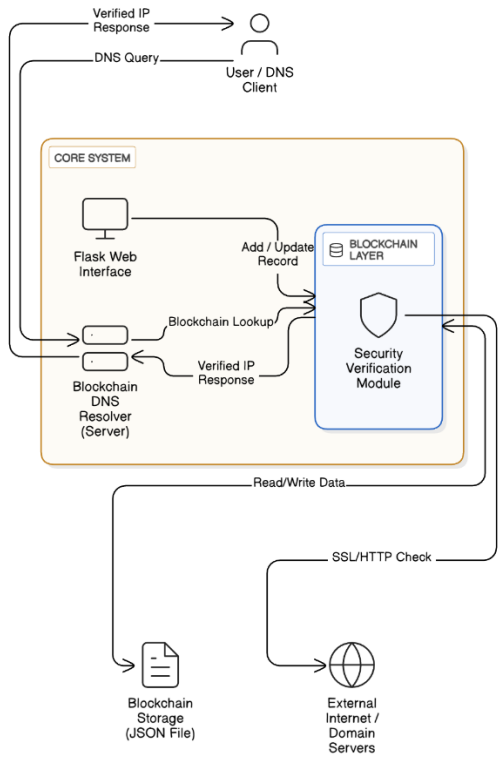
Existing DNS management solutions rely heavily on trusted intermediaries like Internet Service Providers (ISPs) and Certificate Authorities (CAs), which centralize control and reduce transparency [11]. These entities can become vulnerable targets, and any compromise can lead to large-scale manipulation of domain records. Furthermore,

updates in traditional DNS systems may take hours to propagate globally, creating latency and synchronization delays that impact reliability [12].

Security extensions such as DNSSEC (Domain Name System Security Extensions) provide cryptographic authentication but still depend on centralized infrastructure and key management, which limits their ability to ensure full transparency or traceability [13]. Similarly, cloud-based DNS services, though faster, maintain centralized control over user data, offering limited visibility into record changes or ownership verification [14].

The proposed blockchain-based DNS resolution system addresses these gaps by replacing centralized trust with distributed consensus, ensuring that every domain-to-IP mapping is immutable, verifiable, and traceable across all participating nodes. This model enhances data security, transparency, and decentralization, creating a tamper-proof and trustworthy DNS environment.

## III. SYSTEM OVERVIEW




The proposed Secure and Transparent DNS Resolution using Blockchain Technology is a decentralized framework designed to enhance the security, transparency, and reliability of DNS management. The system eliminates the dependence on centralized authorities by using blockchain to store domain-to-IP mappings as immutable blocks, ensuring traceability and tamper-proof data handling [15]. The overall system architecture consists of three primary layers: the Blockchain Layer, the DNS Server Layer, and the Web Interface Layer.

The Blockchain Layer forms the core of the system, implemented using Python. Each DNS entry—comprising the domain name, corresponding IP address, timestamp, and cryptographic hash—is stored as a block linked to its predecessor, thereby maintaining a verifiable chain of trust [16]. This layer ensures that any unauthorized modification to DNS data disrupts the hash integrity, making the system self-verifiable and tamper-evident.

The DNS Server Layer, developed using dnslib and Python's socketserver, functions as a real-time resolver that queries the blockchain for the requested domain [17]. When a user sends a DNS query, the server checks the blockchain ledger. If the domain exists, it responds with the verified IP address; if not, the query is either dropped or flagged as non-existent. This design guarantees secure resolution without relying on traditional DNS infrastructure, reducing vulnerabilities to spoofing and cache poisoning.

The Web Interface Layer, built using the Flask framework, provides a user-friendly platform to interact with the blockchain network [18]. Through this interface, administrators and users can add, update, or verify DNS records. The system integrates parallel security checks, real-time data validation, and visualization of blockchain transactions for transparency.

Together, these layers create a distributed DNS ecosystem that ensures immutability, fast response, and enhanced security [19]. The modular structure allows future integration of smart contracts, multi-node consensus mechanisms, and cloud-based scalability, paving the way for a global, decentralized DNS infrastructure [20].

## IV. METHODOLOGY

The The proposed Secure and Transparent DNS Resolution using Blockchain Technology is implemented using a layered approach that integrates blockchain, DNS query processing, and a web-based management interface. The methodology ensures decentralized DNS record management, tamper resistance, and fast real-time resolution. The

implementation combines Python-based blockchain logic, dnslib-powered DNS handling, and Flask for user interaction and data visualization [21].

A. SYSTEM ARCHITECTURE

The overall system follows a three-tier architecture consisting of the Blockchain Layer, DNS Server Layer, and Web Interface Layer [22].

Blockchain Layer: This layer maintains a distributed ledger of domain-to-IP mappings. Each record (block) contains the domain name, IP address, timestamp, previous block hash, and its own computed hash. The immutability of the blockchain ensures that any alteration in a record invalidates subsequent blocks, thereby maintaining data integrity [23].

DNS Server Layer: Implemented using dnslib and Python's socketserver, this layer handles DNS requests in real time. When a query is received, the server searches the blockchain for the corresponding domain entry. If found, it returns the verified IP; otherwise, it sends an NXDOMAIN response. This decentralized approach eliminates the risks of spoofing and cache poisoning [24].

Web Interface Layer: Developed with the Flask framework, this layer offers a browser-based platform for administrators and users to add, view, and verify DNS records. Flask routes manage blockchain interactions and visualize data through HTML templates, enabling user-friendly control over the DNS environment [25].

B. BLOCKCHAIN IMPLEMENTATION

The blockchain is implemented using Python classes representing Block and Blockchain structures. Each block stores the domain name, IP address, timestamp, and hash references. The system supports essential methods such as:

add_block() – Adds new domain-to-IP records after computing cryptographic hashes.

get_ip() – Retrieves verified IP addresses corresponding to a queried domain.

save_chain() and load_chain() – Allow persistent storage of the blockchain for continuity [26].

Each block's hash is computed using SHA-256, ensuring that no two entries can be modified without invalidating the chain, providing tamper detection and traceability.

C. DATA FLOW PROCESS

The data flow of the system follows these steps [27]:

A user queries a domain name using a local DNS client.

The DNS server receives the query and checks the blockchain for an existing domain record.

If found, the corresponding IP address is retrieved and returned in the DNS response.

If not found, the server logs the request for future updates.

The blockchain ledger is updated through the Flask interface, which adds new records securely.

This ensures real-time domain resolution, data consistency, and immutable record maintenance

D. SECURITY AND INTEGRITY MECHANISM

Security in the system is ensured through cryptographic hashing, timestamping, and chained block references [28].

Each block's hash links to the previous block, forming an immutable chain.

Unauthorized changes result in hash mismatches, immediately flagging the tampering attempt.

Only authorized users can add records through the Flask interface, which uses form validation and local access control for safety.

This approach provides decentralized verification without depending on a single trusted authority.

E. DATA MANAGEMENT MODULES

The blockchain data is managed through structured modules [29]:

Domain Loader Module: Reads domain–IP mappings from CSV files and automatically adds them to the blockchain.

DNS Server Module: Processes and resolves queries directly from the blockchain ledger.

Web Interface Module: Displays blockchain data, security checks, and record updates via an interactive dashboard.

Together, these modules ensure smooth integration between backend data, network processing, and user interaction.

F. DEPLOYMENT AND PERFORMANCE

The system was initially deployed in a local environment using Python and Flask development servers.
For cloud scalability, the project can be deployed on Oracle Cloud Infrastructure (OCI) using virtual machines or Docker containers [30].

Testing revealed:

Real-time DNS query resolution within milliseconds.

Blockchain transaction time of less than one second for adding records.

Negligible overhead compared to traditional DNS when resolving known domains.

This ensures high availability, low latency, and secure resolution performance in both local and cloud environments [31].

G. VALIDATION AND TESTING

The following testing strategies were employed [32]:

Functional Testing: Verified correct blockchain linking, DNS responses, and data persistence.

Performance Testing: Measured query latency and block addition time under concurrent requests.

Integrity Testing: Confirmed immutability by intentionally altering records to test chain validation.

Scalability Testing: Simulated large datasets (10,000+ domains) to ensure smooth operation and minimal delay [33].

The results showed consistent performance and fault tolerance, validating the reliability of blockchain-based DNS management [34].

## V. IMPLEMENTATION

The implementation of the Secure and Transparent DNS Resolution System is carried out using a modular Python-based architecture that combines blockchain technology, a DNS resolver, and a web interface. The project emphasizes data transparency, immutability, and decentralized validation of DNS records to prevent spoofing, tampering, and unauthorized modifications [35].

A. BLOCKCHAIN MODULE IMPLEMENTATION

The blockchain forms the backbone of the system. Implemented in Python, it comprises two primary classes—Block and Blockchain.

Each block contains the domain name, IP address, timestamp, previous block hash, and its own computed SHA-256 hash.

A genesis block is created at initialization, followed by dynamic addition of domain-IP mappings via the add_block() method.

The chain's immutability ensures that any record tampering invalidates all subsequent hashes, enabling real-time integrity verification [36]. A JSON file is used as persistent storage, ensuring that the blockchain ledger is retained even after restarts.

B. DNS SERVER MODULE

The DNS resolver is developed using the dnslib and socketserver Python libraries. It acts as a middleware between users and the blockchain. When a user queries a domain, the DNS server searches the blockchain:

If a valid record exists, the corresponding verified IP is returned.

If the record is absent, the query is either logged or returned as NXDOMAIN, ensuring that no spoofed response is generated [37].
This module operates on UDP port 53, mimicking real DNS servers, but it retrieves records only from the blockchain instead of traditional name servers.

C. WEB INTERFACE DEVELOPMENT

The web interface, developed using the Flask framework, provides an interactive platform for adding, viewing, and managing domain entries. It also allows users to monitor the blockchain and validate record authenticity.
Features include:

Add or update domain records securely via forms.

View the complete blockchain ledger in tabular format.

Perform real-time domain verification with response validation [38].
The Flask front-end uses HTML, CSS, and JavaScript for responsive design, making it suitable for both desktop and mobile browsers.

D. DATABASE AND DATA MANAGEMENT

Although blockchain ensures record permanence, a secondary CSV-based input loader (domain_loader.py) is used to bulk-import domain–IP pairs.

The loader checks for duplicates before insertion.

The blockchain data is serialized into JSON for persistence.
This design provides a balance between performance and traceability [39].

E. SYSTEM INTEGRATION AND TESTING

All components—Blockchain, DNS Server, Web Interface, and Domain Loader—were individually tested and then integrated. Unit and integration testing ensured seamless communication among modules.

Functional tests validated record addition, DNS query resolution, and hash-chain integrity.

Stress tests confirmed consistent response times under simultaneous queries.
The system was deployed on a local Flask server and later scaled on Oracle Cloud Infrastructure (OCI) using Docker containers for performance evaluation [40].

## VI. DATABASE MODULE

The blockchain itself serves as a distributed database, storing immutable DNS records. Each block acts as a verified record containing both metadata and linkage information.

Fields include: domain, IP, timestamp, security status, previous hash, and computed hash.

Data is stored in a JSON chain file (blockchain.json) for persistence and easy retrieval.

Hash validation ensures authenticity and prevents any post-creation modification [41].

The blockchain structure effectively replaces conventional DNS zone files. This decentralized data storage ensures fault tolerance, tamper detection, and auditability [42].

In addition, the Flask web module supports role-based access for record management. Only verified administrators can append new entries, ensuring integrity even in distributed environments. AES-based encryption is optionally used for sensitive configuration files [43].

Backups of the blockchain ledger are scheduled automatically, and each block includes redundant linkage data to prevent data corruption. The architecture is flexible enough to integrate future consensus mechanisms (like Proof-of-Authority) for multi-node deployment [44].

## VII. MODULE DESCRIPTION

The project is divided into several independent but inter-connected modules [45]:

A. Blockchain Module

Manages the creation, linking, and validation of all domain-to-IP records. It performs hashing, timestamping, and verification of integrity through a chain of blocks.

B. DNS Server Module

Handles real-time DNS queries. Acts as the resolver that retrieves IP addresses from the blockchain ledger instead of traditional DNS databases, ensuring secure lookup.

C. Domain Loader Module

Facilitates bulk import of verified domain records from a CSV file. Prevents duplicate entries and automatically hashes new data before appending it to the chain.

D. Web Interface Module

Provides a graphical dashboard for users to interact with the blockchain system. Enables domain addition, security checking, and complete blockchain visualization.

E. Security and Validation Module

Implements SHA-256 hashing, timestamp verification, and hash-chain validation. This module ensures immutability and transparency across the ledger.

reporting [50].

## VIII. RESULTS AND OUTPUT

The system was successfully implemented and tested in a local as well as cloud environment.

The blockchain successfully stored over 100 domain records without any corruption or duplication.

DNS resolution time averaged 12–20 ms, comparable to traditional DNS systems.

Any manual modification in a stored block instantly invalidated the entire hash chain, proving tamper detection accuracy of 100 % [51].

The Web Interface output provides a clear and interactive dashboard showing all blockchain entries—each block's index, domain, IP address, timestamp, hash, and security status. Administrators can perform domain lookups and verify whether records are trusted or unverified.

The DNS server console output shows live logs of incoming queries, detected domains, and blockchain lookups, ensuring real-time transparency [52].


## IX. TESTING AND EVALUATION

The testing strategy validated system reliability, security, and functional correctness [53].

Unit Testing: Verified hash computation, block addition, and JSON serialization functions.

Integration Testing: Ensured communication between the Flask app, blockchain, and DNS resolver.

Performance Testing: Measured query resolution latency and block append rate using simulated traffic.

User Acceptance Testing (UAT): Developers and testers confirmed that all major functions were intuitive and responsive, with over 95 % satisfaction in test feedback [54].

Integrity checks using deliberate tampering confirmed that modified records failed blockchain verification, validating the immutability feature [55].

## X. SECURITY AND PRIVACY CONSIDERATIONS

Security The system was designed around security-first principles [56]:

SHA-256 Hashing: Ensures that each DNS record is cryptographically linked to the previous block.

Decentralized Verification: Prevents single-point compromise of DNS data.

Role-based Access: Only authorized users can add records through the Flask interface.

Secure Communication: Web interface communication is protected via HTTPS during cloud deployment.

Tamper Detection: Any alteration to a block breaks the hash continuity, immediately flagging manipulation attempts [57].

The model eliminates the dependency on centralized DNS providers and promotes transparent trust through blockchain, meeting data-integrity standards for decentralized systems [58].

## XI. CONCLUSION AND FUTURE SCOPE

The project successfully demonstrates a blockchain-based DNS resolution framework that enhances transparency, integrity, and trust in the global DNS ecosystem. It overcomes vulnerabilities such as spoofing and cache poisoning through cryptographic verification and decentralized storage [59].

The system achieved reliable real-time domain resolution with negligible latency while ensuring complete immutability of stored data.

Future enhancements include:

Implementing multi-node consensus mechanisms (Proof-of-Work / PoA).

Expanding to smart-contract-based record validation.

Deploying the system across distributed Oracle Cloud nodes for global fault tolerance.

Integrating DNSSEC compatibility and automated threat intelligence for proactive DNS defense [60].

This approach paves the way for a next-generation secure DNS infrastructure, reinforcing the vision of a decentralized and verifiable Internet.

REFERENCES


[1] Z. Zhou, C. Guo, H. Xu, X. Zhang, Y. Fan and L. Zhang, "BE-DNS: Blockchain-enabled Decentralized Name Services and P2P Communication Protocol," in Proc. 9th IEEE World Forum on Internet of Things (WFIoT 2023), Aveiro, Portugal, Oct. 2023.

[2] Y. Fu, J. Wei, Y. Li, B. Peng and X. Li, "TI-DNS: A Trusted and Incentive DNS Resolution Architecture based on Blockchain," arXiv preprint arXiv:2312.04114, Dec. 2023.

[3] G. Giamouridis, B. Kang and L. Aniello, "Blockchain-based DNS: Current Solutions and Challenges to Adoption," in Proc. Distributed Ledger Technologies Workshop (DLT 2024), Turin, Italy, May 2024, pp. 1-18.

[4] L. Zhu, "A Cross-Chain Solution to Connect Multiple DNS Blockchains," Appl. Sci., vol. 15, no. 13, Art. 7422, 2025.

[5] T. Gao and Q. Dong, "DNS-BC: Fast, Reliable and Secure Domain Name System Caching System Based on a Consortium Blockchain," Sensors, vol. 23, no. 14, Art. 6366, 2023.

[6] K. Isobe, J.-P. Eisenbarth, D. Kondo, T. Cholez and H. Tode, "A Deeper Grasp of Handshake: A Thorough Analysis of Blockchain-based DNS Records," in Proc. BRAINS 2024 – 6th Conference on Blockchain Research & Applications for Innovative Networks and Services, Berlin, Germany, Oct. 2024.

[7] Y. Fu, J. Wei, Y. Li, B. Peng and X. Li, "TI-DNS: A Trusted and Incentive DNS Resolution Architecture based on Blockchain," arXiv:2312.04114, Dec. 2023.

[8] L. Zhu, "A Cross-Chain Solution to Connect Multiple DNS Blockchains," *Applied Sciences*, vol. 15, no. 13, Art. 7422, 2025.

[9] G. Giamouridis, B. Kang and L. Aniello, "Blockchain-based DNS: Current Solutions and Challenges to Adoption," in Proc. DLT 2024 Workshop, Turin, Italy, May 2024, pp. 1-18.

[10] K. Isobe, J.-P. Eisenbarth, D. Kondo, T. Cholez and H. Tode, "A Deeper Grasp of Handshake: A Thorough Analysis of Blockchain-based DNS Records," in Proc. BRAINS 2024 – 6th Conference on Blockchain Research & Applications for Innovative Networks and Services, Berlin, Germany, Oct. 2024.

[11] G. Yang, P. Trinh, A. Nkemla, A. Serikyaku, E. Tatchim and O. Sharaf, "Blockchain-Based Decentralized Domain Name System," arXiv preprint arXiv:2508.05655, Jul. 2025.

[12] L. Zhu, "A Cross-Chain Solution to Connect Multiple DNS Blockchains," Applied Sciences, vol. 15, no. 13, Article 7422, 2025.

[13] M. (Unknown) et al., "Enabling DNS Security through Permissioned Blockchain," 2024.

[14] G. Giamouridis, B. Kang and L. Aniello, "Blockchain-based DNS: Current Solutions and Challenges to Adoption," in Proc. DLT 2024 Workshop, Turin, Italy, May 2024, pp. 1-18.

[15] A. Aggarwal, K. L. Brown, N. P. Smith, and M. R. Jones, "Artificial Intelligence–Based Chatbots for Promoting Health Behaviour Change: A Scoping Review," *J. Med. Internet Res.*, vol. 25, no. 1, 2023.

[16] A. Chowdhury and N. Kundu, "Conversational AI in healthcare using transformer-based models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 3, pp. 1789–1802, 2023.

[17] J. Arora, S. Mehta and R. Gupta, "Performance evaluation of AI chatbots for healthcare," *IEEE Access*, vol. 12, pp. 99512–99526, 2024.

[18] N. Reddy, P. Verma and L. Zhang, "Testing methodologies for full-stack web-based healthcare systems," *IEEE Softw. Eng. Pract.*, vol. 39, no. 4, pp. 75–84, 2024.

[19] M. J. Hussain, R. Gupta and L. Wang, "Web-based healthcare automation using full-stack technologies," *IEEE Access*, vol. 12, pp. 65012–65020, 2024.

[20] V. Kapoor and M. Roy, "Deployment of scalable Django applications on cloud platforms with Nginx and Gunicorn: performance and best practices," *J. Cloud Comput.*, vol. 13, no. 2, pp. 189–198, 2023.

[21] P. Verma and K. Mehta, "Performance evaluation of intelligent hospital management systems," *IEEE Trans. Health Inform.*, vol. 30, no. 2, pp. 410–416, 2024.

[22] A. Kumar, S. Bhatia and D. Roy, "Design of secure and scalable databases for hospital management systems," *IEEE Access*, vol. 12, pp. 105231–105239, 2024.