LHDNS — Ledger-based Hashed Decentralized Naming System

Author: Ahmad Hemmati

Website: https://www.twincodesworld.com

Languages: English

Status: Open Source – Whitepaper | Version: v1.0

Table of Contents

Abstract

1. Introduction

2. Background: Evolution & Limitations of DNS

3. Problem Statement

4. Proposed Solution: LHDNS

5. Architecture Overview

6. System Workflow / Operational Model

7. Security & Threat Model

8. Economic Model & Tokenomics

9. Governance & Community Participation

10. Interoperability & Compliance

11. Performance & Scalability

12. Deployment & Roadmap (Phased)

13. Use Cases

14. Comparative Analysis (brief)

15. Risks & Limitations

16. Conclusion & Call to Action

Related Documents

Abstract

The Domain Name System (DNS), while foundational to the global Internet, exhibits structural weaknesses: centralized trust anchors, plaintext metadata exposure, and long-lived records that facilitate surveillance, censorship, and targeted attacks. LHDNS (Ledger-based Hashed Decentralized Naming System) reimagines name resolution by combining cryptographic hashing, rotating nonces, ephemeral ledger entries, and decentralized consensus. Identifiers are derived from secure hashes; lookup events are time-limited and verifiable; and identity is decoupled from transport. LHDNS provides a privacy-preserving, censorship-resistant, and verifiable naming layer while offering feasible migration via gateways, browser APIs, and dual-stack compatibility. This whitepaper details the architecture, protocol flows, security model, governance, tokenomics, and deployment roadmap for LHDNS.

1. Introduction

The Domain Name System (DNS), while foundational to the Internet and a critical infrastructural component, was designed for a different era. Its hierarchical, authority-driven model reflects early assumptions about trust and scale that do not hold for today's adversarial, privacy-sensitive, and massively distributed environment. LHDNS proposes a clean-slate re-architecture: ephemeral, ledger-backed resolution events; hash-derived identifiers with rotating nonces; and decentralized verification via a permissionless ledger. The goal is to provide name resolution that is verifiable, unlinkable, censorship-resistant, and deployable in phases alongside legacy DNS.

2. Background: Evolution & Limitations of DNS

DNS solved a scalability problem by replacing static hosts files with a hierarchical naming system. Over decades it scaled to billions of devices, but its design choices now create systemic weaknesses:

- Authority concentration in root and TLD operators.
- Plaintext queries exposing user intent.
- Long-lived caches and records enabling tracking and cache-poisoning spread.
- Centralized resolvers becoming points of jurisdictional or corporate control.
  Existing mitigations (DNSSEC, DoH/DoT) improve integrity or confidentiality in parts but do not remove centralized control or protect against traffic-pattern analysis and censorship.

3. Problem Statement

DNS's hierarchical model and centralized resolvers produce multiple failures: single points of failure (DoS or misconfiguration), censorship (ISP/government filtering/hijacking), privacy loss (query profiling), spoofing and MitM risk, and scalability limitations for real-time IoT workloads. A fundamentally different trust model—one based on distributed consensus, ephemeral state, and cryptographic verification—is required.

4. Proposed Solution: LHDNS

LHDNS is a ledger-based, hash-driven naming system built on these core ideas:

- **Hashed identifiers**: Clients and services use cryptographic hashes bound to rotating nonces so tokens cannot be linked across sessions.
- **Ephemeral ledger entries**: Resolutions are recorded as short-lived ledger events with strict TTLs; entries expire and are pruned.
- **Off-ledger data transfer**: After authentication, actual content transfer uses existing transport (TCP/IP + TLS) off-ledger.
- **Decentralized consensus**: A permissionless ledger distributes authority; validator nodes validate entries and maintain integrity.
- **Migration path**: Gateways and browser APIs enable interoperability with DNS during rollout.

Design principles: decentralization of trust, privacy & unlinkability, cryptographic integrity, ephemeral state, censorship-resistance, scalability, deployability, and open governance.

5. Architecture Overview

Components:

- **Client/Browser**: Generates session-specific tokens (hash(client_id + server_id + nonce)), submits ledger lookup requests, verifies signed responses.
- **Service / Website**: Publishes signed service descriptors (public keys, transports, nonce policy) and validates incoming hashed requests.
- **Ledger / Validator Nodes**: Record ephemeral lookup events, validate signatures, prune expired entries, and participate in consensus.
- **Gateways / Resolvers**: Bridge to legacy DNS and aid bootstrapping and migration.
- **Relays / Privacy Layer**: Optional multipath relays or onion-like forwarding for additional unlinkability.

High-level query lifecycle:

1. Client forms a hash-token with nonce → broadcasts to ledger.
2. Validator nodes validate and index the token.
3. Service checks for matching token and publishes signed confirmation.
4. Client verifies signature → establishes off-ledger secure connection.
5. Ledger entry expires after TTL.

6. System Workflow / Operational Model

Phases described succinctly:

- **Initial bootstrap**: Browser search-engine or gateway assisted to obtain initial service descriptor and initial nonce exchange.
- **Lookup**: Client constructs hashed token (incorporating nonce) and submits to ledger (or to peer nodes which gossip it to ledger).
- **Validation**: Validator nodes confirm structure, signature bindings, and freshness; return proof of existence.
- **Connection**: Off-ledger, mutually authenticated TLS session; site issues new nonce for next session upon completion.
- **Expiration**: Ledger enforces TTL and prunes expired tokens.

Appendix A contains module-level pseudocode and exact message formats (hash algorithms, signature schemes, nonce lifecycles).

7. Security & Threat Model

Objectives: confidentiality of intent, integrity of responses, availability, unlinkability, and censorship resistance.

Threats considered: passive eavesdroppers, traffic-pattern analysis, MitM, replay attacks, Sybil/eclipsing, DoS on nodes, governance capture.

Mitigations:

- Hash + nonce design prevents linkage and replay.
- Digital signatures and ledger immutability harden against spoofing.
- Staking & slashing reduce Sybil incentives.
- Rate limiting, PoW/light anti-spam measures, and adaptive fees mitigate DoS.
- Fallback/gateway patterns and emergency governance (time-locked upgrades) handle catastrophic failures.

DNSSEC's limited adoption and centralized root trust are noted: LHDNS moves trust to a decentralized consensus that avoids single root anchors.

8. Economic Model & Tokenomics

Core components:

- **Native token (LHD)**: transaction fees, staking collateral, reward distribution, governance voting.
- **Fee model**: minimal per-lookup fees to deter spam; dynamic adjustment under congestion.
- **Staking & validator rewards**: nodes stake LHD to participate and earn fees; misbehavior triggers slashing.
- **Treasury**: a portion of fees funds audits, grants, and ecosystem growth.

- **Sustainability**: fee/issuance parameters balanced to ensure low cost for typical users (including IoT) while sufficiently rewarding validators.

Formal reward/slash formulas and micropayment channel designs are in Appendix A.

9. Governance & Community Participation

Governance is on-chain and transparent:

- Token-weighted voting with anti-capture mechanisms (delegation, quadratic tweaks, capped influence).
- Proposal lifecycle: draft → discussion → on-chain vote → time-locked execution.
- Treasury-managed funding for audits, developer grants, and public goods.
- Community councils or technical committees for rapid-response and security oversight.

10. Interoperability & Compliance

- Gateways support DNS ↔ LHDNS translation.
- TLS/PKI and DID integration for backwards-compatible authentication paths.
- Compliance gateways for enterprises/regulators that need logging under explicit opt-in, without weakening the core privacy model.
- Standards engagement (IETF) planned for long-term compatibility.

11. Performance & Scalability

- Two-layer ledger idea: fast ephemeral gossip layer for lookup events + slower governance/audit layer for signed digests.
- Gossip with adaptive fanout, selective subscriptions, in-memory TTL pruning to keep latency low.
- Expected latency targets: interactive lookups in sub-second to low-second range under normal network conditions.
- IoT support: lightweight nodes and gateway proxies to offload heavy functions.

12. Deployment & Roadmap (Phased)

Phase 0 (0–6 months): formal spec, reference implementation prototypes, academic/industry review.
Phase 1 (6–12m): private testnet, security audits, SDKs.
Phase 2 (12–18m): public testnet, incentivized participation, gateway plugins.
Phase 3 (18–24m): pilot deployments with browsers/hosters, treasury & early governance.
Phase 4 (24–36m): mainnet launch, staking & full governance.
Phase 5 (36m+): ecosystem expansion, cross-chain interoperability, post-quantum upgrades.

KPIs and acceptance criteria (stability, latency, decentralization index, economic validation) are defined per phase (see Appendix A).

13. Use Cases

- Censorship-resistant browsing (users in restricted jurisdictions).
- Privacy-first consumer apps and messaging services.
- Secure IoT discovery with lightweight lookups.
- Enterprise private naming with opt-in compliance.
- Web3 and decentralized services discovery (DApps, storage, identity).
- Critical infrastructure resilient naming.

14. Comparative Analysis (brief)

- DNS vs DNSSEC vs DoH/DoT vs blockchain naming: LHDNS targets a combination of privacy, practical deployability, and ledger-backed verification with ephemeral state — addressing the gaps left by each existing approach.

15. Risks & Limitations

- Bootstrapping decentralization and preventing early centralization.
- Balancing privacy vs latency trade-offs.
- Regulatory pressure in some jurisdictions (mitigated by enterprise opt-in gateways).
- Economic parameter tuning to avoid over/under-incentivizing nodes.

16. Conclusion & Call to Action

LHDNS is a practical rethinking of naming — combining ledger-backed ephemeral resolution, hashed identifiers with rotating nonces, and an aligned economic/governance model. We invite researchers, developers, operators and funders to collaborate: implement reference clients, run validators on testnets, audit the protocol, and build gateways and SDKs. Together we can build a privacy-preserving, censorship-resistant naming layer for the next-generation Internet.

Related Documents

- Appendix A – Technical Modules
- Annex A – Extended Technical Report
- Appendix B – Glossary & Definitions
- Appendix C – References & Related Work
- Appendix D – Threat Scenarios & Attack Trees