DWeb Server: Product Requirements Document

Version: 2.3
Date: Nov 6, 2025
Project Code Name: DWeb Server
Status: Draft for Review

Executive Summary

DWeb Server is an anonymous publishing platform built on the Open Index Protocol (OIP) that enables whistleblowers, journalists, researchers, and activists to publish information that cannot be censored, while maintaining complete anonymity and ensuring persistent discoverability.

The system combines **one-click home-network deployment**, **Tor onion routing for writer privacy**, **WordPress for familiar authoring**, **persistent peer-to-peer distribution**, and **decentralized name resolution** to create a resilient publishing platform with no single point of failure and no monthly hosting fees.

Core Promise: Publish anonymously. Index persistently. Distribute widely. Discover reliably. Moderate locally. Resolve names without central control.

Key Differentiators

- **Anonymous Publishing**: Tor onion routing hides publisher identity and location
- **Persistent Index**: Records remain discoverable even when original publishers go offline
- **Flexible Registration**:
    - **Run your own node**: No registration required at all—full self-sovereignty
    - **Use existing gateway**: Simple registration (username + password only, no email verification)
    - **Your choice**: Trade convenience for maximum control
- **Gateway-Scoped Moderation**: Each node operator decides what to index and serve
- **Decentralized Name Resolution**: ENS for mainstream reach, GNS for private lookups
- **Local AI/RAG**: On-device question answering over your indexed content (Alfred)
- **Location Agnostic**: DID spec adherence means data can live anywhere (any blockchain, any storage)
- **One-Click Deploy**: Raspberry Pi images and Docker Compose for home networks
- **No Monthly Fees**: Self-hosted with peer-to-peer distribution

Two Deployment Models

DWeb Server offers two deployment paths, each optimized for different user needs:

Model 1: Self-Hosted Node (Maximum Sovereignty)

Who it's for: Users who want complete control and zero trust dependencies

Setup:

- Flash Raspberry Pi image or run Docker Compose
- Generate DID from 12-word mnemonic during setup
- No registration, no account, no server to trust
- Full node runs on your hardware

Benefits:

- ✅ **Zero registration**: Generate keys and publish immediately
- ✅ **No trusted parties**: Your keys never leave your device
- ✅ **Full control**: You decide what to index and serve
- ✅ **Maximum privacy**: No external party knows your identity

Trade-offs:

- ⚠️ Requires technical setup (though simplified with wizard)
- ⚠️ Hardware costs ($35-200 one-time)

- ⚠️ Your responsibility to backup mnemonic

Use Cases:

- Whistleblowers requiring maximum anonymity
- Organizations running their own infrastructure
- Privacy advocates and technical users
- High-stakes publishing scenarios

Model 2: Gateway Registration (Maximum Convenience)

Who it's for: Users who want simplicity and multi-device access

Setup:

- Visit existing gateway URL
- Register with username + password (no email)
- Gateway generates and stores encrypted mnemonic
- Publish from any device

Benefits:

- ✅ **Zero setup**: No hardware, no installation
- ✅ **Multi-device**: Access from phone, laptop, anywhere
- ✅ **No hardware costs**: Free to start publishing
- ✅ **Automatic backups**: Gateway maintains encrypted mnemonic

Trade-offs:

- ⚠️ Trust gateway operator with encrypted keys
- ⚠️ Username + password required (though no email)
- ⚠️ Gateway could be shut down (migrate to self-hosted)

Use Cases:

- Journalists needing quick publishing workflow
- Researchers sharing non-sensitive findings
- Activists coordinating across multiple locations
- First-time users exploring the platform

Migration Between Models

Gateway → Self-Hosted:

1. Export your 12-word mnemonic from gateway
2. Set up your own DWeb Server node
3. Import mnemonic during setup
4. Same DID, same identity, now fully self-sovereign

Self-Hosted → Gateway:

1. Export your 12-word mnemonic from your node
2. Register on gateway with username + password
3. Import mnemonic (gateway encrypts and stores it)
4. Same DID, now accessible from multiple devices

Key Point: Your DID identity is portable. The deployment model is a choice of convenience vs. control, not a permanent commitment.

Vision & Mission

Vision Statement

"A world where truth cannot be silenced, where publishers cannot be identified against their will, and where information persists beyond the reach of censorship."

Mission Statement

"Build the simplest, most resilient platform for anonymous publishing that combines the familiar (WordPress authoring) with the revolutionary (DID-based identity, Tor anonymity, and persistent peer distribution)—enabling anyone to publish information that matters without fear."

The DWeb Server Name

DWeb Server emphasizes what it is: a decentralized web server that anyone can run. The name is: - **Descriptive**: Clearly communicates purpose (server for the decentralized web) - **Accessible**: No learning curve to understand what it does - **Historical**: Connects to the DWeb movement started at the 2016 DWeb Summit - **Inclusive**: Appeals to technical and non-technical users alike

---

Background & Historical Context

The DWeb Summit Legacy (2016-Present)

DWeb Server represents the culmination of nearly a decade of decentralized web development, starting with the first **DWeb Summit in 2016** organized by the Internet Archive. That historic gathering brought together key technologies that form the foundation of this project:

The Five Pillars (2016):

1. **OIP/DLOA (Decentralized Library of Alexandria)**: Blockchain-indexed metadata combined with distributed file networks.
2. **IPFS (InterPlanetary File System)**: Content-addressed storage and peer-to-peer file distribution, with core projects spinning out (multiformats, libp2p, IPLD).
3. **WebTorrent**: Browser-native BitTorrent via WebRTC data channels, bringing P2P streaming to web applications without plugins.
4. **GUN**: Decentralized graph database with offline-first CRDT replication and cryptographic user identity.
5. **DIDs (Decentralized Identifiers)**: Self-sovereign identity concepts crystallizing in Rebooting-the-Web-of-Trust and W3C Credentials Community Group work.

Evolution Across Three Eras (2016-2025):

Era 1: Foundation & Formalization (2016-2019)

- **DWeb Summit 2016** convenes all five technologies at Internet Archive; Tim Berners-Lee calls DLOA "thrilling"
- **OIP/DLOA**: Demonstrated blockchain-indexed library (2016) → formal OIP specification and Working Group (2017) → Caltech ETDB scientific data deployment (2018) → Teton County blockchain land records and DWeb Summit presentation (2019)
- **IPFS**: Core projects spin out—multiformats, libp2p, IPLD (2016) → Filecoin token sale funds development (2017) → Internet Archive demos multi-transport DWeb using IPFS, GUN, WebTorrent (2018) → Kubo hardening with transport/security upgrades (2019)
- **WebTorrent**: Desktop app launched via Electron (2016) → highlighted as browser-native torrenting via WebRTC (2017) → bridges WebRTC browser peers to TCP/UDP swarms (2018) → community adoption for in-browser streaming (2019)
- **GUN**: Matures as offline-first CRDT database (2016-2017) → featured in Internet Archive's multi-transport DWeb demo (2018) → positioned as JS graph DB with multi-master replication (2019)
- **DIDs**: Concept development in Rebooting-the-Web-of-Trust (2016) → W3C discussions advance standards (2017-2018) → W3C DID Working Group formally chartered (2019)

Era 2: Standards & Transformation (2020-2022)

- **OIP**: Professional video (Streambed) and news publishing (Al Bawaba) deployments (2020) → Web Monetization integration and Consensus Conference demos with token-gated video (2021) → complete Arweave rewrite with modern API architecture and HD wallet authentication (2022)
- **IPFS**: Filecoin mainnet launches as permanence layer (2020) → performance and routing research advances (2021) → web-scale retrieval and decentralization research (2022)
- **WebTorrent**: Stable beta v0.24.0 with broad media support (2020) → mature stable releases across desktop and browser (2021-2022)
- **GUN**: Continued community deployment in P2P and decentralized applications (2020-2022)
- **DIDs**: Specification advances through W3C Working Group process (2020-2021) → **W3C Recommendation published (DID Core v1.0) on July 19, 2022**—official standard achieved

Era 3: Integration & Production (2023-2026)

- **OIP**: Alexandria Reference Client with Elasticsearch integration (2023 & 2024) → open-source AI model integration (2025) → **GUN, BitTorrent/WebTorrent, and IPFS fully integrated**; FitnessAlly AI-powered end-user app for managing health & fitness (2025)
- **IPFS**: Bitswap/Graphsync performance optimization and Interplanetary Shipyard web integration (2023) → post-gateway direct retrieval research (2024) → web-native retrieval advances (2025)
- **WebTorrent**: Stable ecosystem with continued browser and desktop adoption (2023-2025)
- **GUN**: Community maintenance (2023-2024) → **real-time sync integrated into OIP for distributed applications** (2025)
- **DIDs**: Follow-on documentation for DID Resolution and Extensions (2023) → interoperability refinements (2024) → widespread adoption as identity standard (2025)
- **DWeb Server**: **Unifies all five pillars into single zero-dependency product** (early 2026)

What Makes This Different:

Previous efforts required choosing between technologies or integrating them manually. DWeb Server is the first product to unify **all five DWeb technologies**—OIP, IPFS, WebTorrent, GUN, and DIDs—plus modern additions (Tor, WordPress, AI) into a cohesive system with **zero central points of failure**.

Why This Project Matters Now

Rising Censorship: - 40+ major lawsuits against Internet Archive by media companies - UK age verification mandates - Russian takedown demands and DNS blocking - Great Firewall of China expansions - EU copyright reexamination (June 2026)

Attacks on Infrastructure: - Internet Archive DDoS attacks (repeated) - October 2024 intrusion (2 weeks before election) - Cloudflare's gatekeeper business model (charge to access websites)

Need for Anonymous Publishing: - Whistleblowers exposing government/corporate wrongdoing - Researchers disrupting entrenched industries - Journalists in authoritarian regimes - Activists organizing under surveillance

Brewster Kahle's Vision

From the DWeb Summit session and subsequent discussions:

> "Censorship resistant WebServer. Easy and fun, no monthly fee, open source, private, reliable. One-click downloadable software package to build your own webserver that protects writer privacy."

Required Components: 1. **Tor onion server** for writer privacy 2. **WordPress** for familiar authoring 3. **Decentralized DNS** for name resolution 4. **Automatic backups** to archive.org for reliability

Key Quote from Freedom of the Press Foundation: "We need this!"

Core Principles

1. Anonymity Without Compromise

- Publisher IP addresses never exposed via Tor onion routing
- **Two registration models**:
  - **Self-hosted node**: No registration required—full control, full sovereignty
  - **Existing gateway**: Username + password only (no email, no personal information)
- DID-based identity without revealing real-world identity
- Your keys, your identity, your choice of deployment

2. Persistent Discoverability

- Records indexed in an open, queryable database
- Metadata survives even when publishers go offline
- Multiple gateways replicate index for redundancy
- Discovery works when any single node fails

3. Gateway-Scoped Moderation

- Each operator decides what their gateway indexes
- No global censorship authority
- Abuse controls at ingest and index layers
- Publisher anonymity preserved during review

4. Location Agnostic by Design

- Records stored anywhere: blockchain, distributed hash tables, local storage
- DID spec adherence means true portability
- Currently uses Arweave; could use Bitcoin, Ethereum, or any other chain
- Change backend without changing application layer

5. No Single Choke Point

- Multiple gateways can accept submissions
- Peer-to-peer content distribution
- No central registry or single point of failure
- Each node is independently operated

6. Simple and Accessible

- Familiar WordPress authoring experience
- One-click installation (Raspberry Pi images, Docker Compose)
- No monthly fees or cloud dependencies
- Works on home networks

7. End-to-End Verifiable

- DID-based signing proves authorship
- Cryptographic proofs without revealing identity
- Key rotation and revocation supported
- Verifiers check provenance from index

User Stories & Personas

Persona 1: The Whistleblower

Background: Government analyst with evidence of wrongdoing
Needs: Publish documents anonymously without being identified
Fears: Network surveillance, metadata analysis, retaliation

User Story:

> "As a whistleblower, I want to publish sensitive documents to a platform that cannot reveal my identity or location, so that I can expose wrongdoing without endangering myself or my family."

Acceptance Criteria: - Can run own DWeb Server node with no registration (full sovereignty) - Alternatively, can register on existing gateway with username + password only - Can submit documents via Tor without revealing IP address - Can sign documents with a DID key not linked to real identity - Documents remain accessible even if original gateway is shut down - Cannot be traced through timing analysis or network forensics - Can export mnemonic and migrate between self-hosted and gateway modes

Persona 2: The Independent Journalist

Background: Freelance reporter covering corporate corruption
Needs: Publish articles that cannot be taken down by legal threats
Fears: SLAPP suits, content takedowns, loss of platform

User Story:

> "As an independent journalist, I want to publish my investigative reporting to a platform where it cannot be censored or removed, so that my work remains accessible regardless of legal pressure."

Acceptance Criteria: - Can write articles in WordPress with familiar interface - Can publish to persistent index that no single entity controls - Articles discoverable through open search interface - Can build reputation through pseudonymous DID without doxxing

Persona 3: The Academic Researcher

Background: University researcher with findings that threaten pharmaceutical industry
Needs: Publish research data that contradicts corporate-funded studies
Fears: Pressure on university, loss of funding, career damage

User Story:

> "As a researcher, I want to publish my findings in a way that ensures they cannot be suppressed, so that the scientific community has access to unbiased data."

Acceptance Criteria: - Can publish datasets with proper provenance - Can attach cryptographic signatures proving data integrity - Data remains available even if university demands takedown - Can publish under pseudonym while maintaining scientific credibility

Persona 4: The Gateway Operator

Background: Privacy advocate running a node from home
Needs: Contribute to censorship-resistant infrastructure without liability
Fears: Legal trouble for hosting objectionable content

User Story:

> "As a gateway operator, I want to run a node that helps publishers while retaining control over what I index and serve, so that I can support free speech without taking on unlimited legal risk."

Acceptance Criteria: - Can set moderation policies for my gateway - Can approve/quarantine/reject submissions based on my criteria - Can operate node from home network without exposing personal info - Not liable for content indexed by other gateways

Persona 5: The Archivist/Reader

Background: Historian researching suppressed information
Needs: Discover and access records that might be censored elsewhere
Fears: Content disappearing before it can be studied

User Story:

> "As a researcher, I want to search across multiple gateways to find records that might be suppressed on mainstream platforms, so that I can access comprehensive historical information."

Acceptance Criteria: - Can query across multiple gateway indexes - Can verify provenance and authenticity of records - Can use local AI (Alfred) to ask questions about discovered content - Content remains accessible even if original publishers are offline

---

System Architecture

High-Level Architecture Diagram

```
┌─────────────────────────────────────────────────────────────────┐
│                         PUBLISHER LAYER                         │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌──────────────┐     ┌─────────────┐     ┌──────────────┐    │
│  │ WordPress    │──┬─>│  Tor Onion  │──┬─>│   Gateway    │    │
│  │ (Authoring)  │  │  │   Service   │  │  │  (Publish    │    │
│  │              │  │  │             │  │  │   Ingest)    │    │
│  └──────────────┘  │  └─────────────┘  │  └──────────────┘    │
│                    │                    │                       │
│  ┌──────────────┐  │  ┌─────────────┐  │  ┌──────────────┐    │
│  │  Air-Gapped  │──┘  │Export .lopkg│──┘  │   Status     │   │
│  │   Signing    │     │  (Offline)  │     │   API        │    │
│  └──────────────┘     └─────────────┘     └──────────────┘    │
└─────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌─────────────────────────────────────────────────────────────────┐
│                      PERSISTENCE LAYER                          │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌──────────────┐     ┌─────────────┐     ┌──────────────┐    │
│  │  Arweave     │     │    GUN      │     │ Alternative  │    │
│  │ (Metadata    │     │ (Private    │     │  Blockchain  │    │
│  │  Storage)    │     │  Records)   │     │  (Optional)  │    │
│  └──────────────┘     └─────────────┘     └──────────────┘    │
│                                                                 │
│  Note: DID spec means metadata can live anywhere              │
└─────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌─────────────────────────────────────────────────────────────────┐
│                     DISTRIBUTION LAYER                          │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌──────────────┐     ┌─────────────┐     ┌──────────────┐    │
│  │  BitTorrent  │     │    IPFS     │     │    HTTP      │    │
│  │ (WebTorrent) │     │  (Optional) │     │   Fallback   │    │
│  └──────────────┘     └─────────────┘     └──────────────┘    │
│                                                                 │
│  Peer-to-peer distribution reduces reliance on publishers     │
└─────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌─────────────────────────────────────────────────────────────────┐
│                       DISCOVERY LAYER                           │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌──────────────┐     ┌─────────────┐     ┌──────────────┐    │
│  │Elasticsearch │     │   Gateway   │     │   Gateway    │    │
│  │   (Index)    │     │   Node A    │     │   Node B     │    │
│  │              │     │  (Moderated)│     │  (Moderated) │    │
│  └──────────────┘     └─────────────┘     └──────────────┘    │
│                                                                 │
│  Each gateway independently indexes and serves records        │
└─────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌─────────────────────────────────────────────────────────────────┐
│                       CONSUMER LAYER                            │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌──────────────┐     ┌─────────────┐     ┌──────────────┐    │
│  │   Web UI     │     │   Alfred    │     │   Mobile     │    │
│  │  (Browser)   │     │ (Local AI)  │     │   Apps       │    │
│  │              │     │   RAG Q&A   │     │              │    │
│  └──────────────┘     └─────────────┘     └──────────────┘    │
│                                                                 │
│  Readers discover and verify content across gateways          │
└─────────────────────────────────────────────────────────────────┘
```

Component Breakdown

1. Publisher Layer (New for DWeb Server)

WordPress Plugin ("LO Publisher") - Familiar authoring interface with Gutenberg editor - Local DID key management (ed25519 signing keys) - Export signed packages (.lopkg files) for offline relay - Direct submission to gateway via Tor onion service - Media manifest generation (no large files over Tor)

Tor Onion Service - v3 hidden service for publisher anonymity - Accepts publish submissions on loopback-only gateway - Strips all identifying headers and metadata - Batches submissions to reduce timing correlation

Air-Gapped Signing - Create signed records offline - Export to USB drive or courier - Submit from different network/location - Complete separation of signing and submission

2. Persistence Layer (Blockchain-Agnostic)

Current Implementation: Arweave - Permanent storage of record descriptors - Immutable metadata cannot be altered - Transaction IDs provide content addressing

DID Spec Adherence - Records use `did:arweave:txid` format - Can change to `did:bitcoin:...` , `did:ethereum:...` , etc. - Persistence mechanism is abstracted from application layer - Metadata structure remains identical regardless of backend

Alternative Storage Options - Bitcoin blockchain (OP_RETURN or Ordinals) - Ethereum (IPFS hash + ENS) - Local DHT (Distributed Hash Table) - Any system supporting DID resolution

3. Distribution Layer (Content Delivery)

BitTorrent/WebTorrent - Peer-to-peer distribution for large files - Magnet URIs in record metadata - Browser-compatible via WebTorrent - Persistent seeding by gateway nodes

IPFS (Optional) - Decentralized content addressing - CID references in record metadata - Gateway pinning for availability

HTTP Fallback - Direct serving from gateway nodes - Range request support for streaming - Authentication for private records

Key Principle: Content is **referenced** in records, not embedded. Large files never go over Tor.

4. Discovery Layer (Gateway Nodes)

Gateway Node Architecture - Elasticsearch for full-text search - GUN for real-time synchronization - Moderation queue with configurable policies - Public API for queries and retrieval

Gateway Moderation - Each operator sets own policies - Size limits, MIME type allowlists - Tag-based auto-approval/quarantine - Hash-based deny lists - Rate limiting per publisher DID

Cross-Gateway Discovery - Records replicated across multiple gateways - Each gateway has independent moderation - Content suppressed on one gateway may appear on others - No global consensus required

5. Consumer Layer (Discovery and Retrieval)

Web UI - Browse and search across gateway indexes - View records with full provenance - Verify DID signatures - Download content via peer networks

Alfred (Local AI/RAG) - On-device question answering - Queries your local gateway's indexed content - No cloud dependencies - Voice interface option (Mac client)

Mobile/Desktop Apps - Native applications for iOS/Android/macOS/Windows - Offline reading with synchronized indexes - Peer-to-peer content retrieval

---

Key Features

Feature 1: Anonymous Publishing via Tor

Description: Publishers submit records through Tor onion services, ensuring network-level anonymity.

User Benefit: Whistleblowers and activists can publish without revealing their IP address, location, or network path.

Technical Implementation: - Tor v3 onion service with persistent keys - Gateway bound to loopback only (127.0.0.1) - All headers stripped (X-Forwarded-For, Via, etc.) - Batched submission to clearnet backends (timing obfuscation)

Acceptance Criteria: - Gateway accessible via .onion address - No IP addresses logged from onion submissions - Tor Browser successfully publishes test records - Timing analysis reveals no one-to-one correlation

Feature 2: WordPress-First Authoring

Description: Familiar WordPress interface for composing posts, pages, and media-rich content.

User Benefit: No learning curve—journalists and writers use tools they already know.

Technical Implementation: - WordPress plugin ("LO Publisher") - Gutenberg sidebar panel for OIP integration - Field mapping: WP fields → OIP template fields - Local key management with encrypted storage

Acceptance Criteria: - Standard WordPress install + plugin works - Compose post → click "Publish to OIP" → receive receipt - Export signed package works offline - Media references generated automatically

Feature 3: DID-Based Identity (OIP 0.9)

Description: Publishers use Decentralized Identifiers (DIDs) based on HD wallet cryptography, following W3C standards.

User Benefit: - **Run your own node**: No registration required—generate DID from mnemonic, publish immediately - **Use existing gateway**: Simple registration (username + password) to store encrypted mnemonic - Either way: No email, no personal information, just cryptographic identity you control

Technical Implementation: - BIP-39 mnemonic (12-word recovery phrase) - BIP-32 HD wallet with SLIP-0043 derivation paths - Purpose-scoped child keys (signing, delegation, revocation) - DID Document records published to index

Derivation Path Structure: ``` m / 43' / 176800' / / /

Where: - 43' = SLIP-0043 custom purpose (not SLIP-0044 coin type) - 176800' = OIP namespace - sub-purpose' ∈ {0=sign, 1=encrypt, 2=delegate, 3=revoke, ...} - account' = account number (0' for default) - index = per-record key (derived from txId for xpub mode) ```

Key Security Features: - Master key never used for signing - Read-only xpub allows verification without private key - Rolling keys: new key burns old keys automatically - Binding proofs for hardened child keys

Acceptance Criteria: - Generate DID from mnemonic (no registration if self-hosting) - Optional registration on gateway stores encrypted mnemonic - Sign records with child keys derived from mnemonic - Verify signatures via xpub derivation - Export/import mnemonic works across nodes - Self-hosted and gateway-registered users have identical capabilities

Feature 4: Gateway-Scoped Moderation

Description: Each gateway operator decides what to index and serve—no global censorship authority.

User Benefit: Publishers aren't at mercy of single moderator; content suppressed on one gateway may appear on others.

Technical Implementation: - Configurable ingest policies per gateway - MIME allowlists, size limits, tag rules - Review queue with approve/quarantine/reject actions - Hash denylists (CSAM, malware)

Moderation Controls: - **Size caps**: Max record body size (configurable, default 25MB metadata) - **MIME allowlists**: Image/video/document types allowed - **Tag policies**: Auto-approve records with certain tags - **Rate limits**: Per-publisher DID throttling - **Hash denylists**: Known abusive content blocked

Acceptance Criteria: - Operator sets policy → records filtered accordingly - Reject reasons returned to publisher with error codes - Approved records appear in index within SLA - Different gateways show different result sets

Feature 5: Persistent Discoverability

Description: Records remain discoverable even when original publishers go offline.

User Benefit: Information survives publisher shutdown, network failure, or legal pressure.

Technical Implementation: - Metadata stored on persistent blockchain (Arweave) - Content distributed via peer networks (BitTorrent, IPFS) - Multiple gateway nodes replicate index - Elasticsearch provides full-text search

Discovery Flow:
```
1. Publisher submits record → stored on Arweave 2. Gateway indexes record → Elasticsearch 3. Other gateways sync record → replicate ind
```

Acceptance Criteria: - Publish record → gateway indexes within 60 seconds - Other gateways sync within 5 minutes - Publisher node goes offline → content still discoverable - Peer network serves content without publisher seeding

Feature 6: Local AI/RAG (Alfred)

Description: On-device AI assistant that answers questions about indexed content without cloud dependencies.

User Benefit: Query your gateway's content privately; no data sent to external services.

Technical Implementation: - Local LLM (Ollama: llama3.2:3b or similar) - RAG pipeline with Elasticsearch vector search - Voice interface option (Mac client) - Conversation history stored locally (encrypted)

Alfred Capabilities: - "Who published research on topic X?" - "What records mention company Y?" - "Summarize all articles by author Z" - "Find datasets related to keyword W"

Acceptance Criteria: - Install gateway → Alfred works out of box - Ask natural language question → receive relevant records - Voice input/output works on Mac client - No network calls to external AI services

Feature 7: Create-But-Don't-Send

Description: Export signed records as files for out-of-band relay—complete air gap between signing and submission.

User Benefit: Maximum anonymity—sign on one machine, submit from unrelated network.

Technical Implementation: - WordPress plugin exports `.lopkg` (DWeb Server Package) - Package contains: signed record JSON, media manifest, optional attachments - Separate tool/endpoint accepts `.lopkg` submissions - Courier or automated relay forwards to gateway

Use Cases: - Journalist signs on air-gapped laptop - Courier carries USB drive across border - Third party submits from coffee shop WiFi - Complete separation of identity and network

Acceptance Criteria: - Export signed package from WordPress - Package validates offline - Submit package via separate network - Gateway accepts package and indexes record

Feature 8: One-Click Deployment

Description: Pre-configured images for Raspberry Pi and Docker Compose for zero-config setup.

User Benefit: Non-technical users can run a gateway from their home network.

Technical Implementation: - **Raspberry Pi Image**: Flash SD card → boot → follow wizard - **Docker Compose**: `docker-compose up` → entire stack running - **Setup Wizard**: Configure DID keys, moderation policies, peers

Stack Includes: - WordPress with LO Publisher plugin - Tor onion service (auto-configured) - Gateway node (OIP backend) - Elasticsearch (index layer) - MediaSeeder (BitTorrent/IPFS) - Alfred (local AI/RAG)

Acceptance Criteria: - Raspberry Pi boots and presents setup wizard - Docker Compose starts all services - Publish test record end-to-end within 10 minutes - Query via Alfred works immediately

Feature 9: Verifiable Provenance

Description: Cryptographic proofs allow readers to verify who published what, and when.

User Benefit: Trust information without trusting intermediaries—verify signatures directly.

Technical Implementation: - Each record signed with publisher's DID child key - Signature algorithm: Ed25519 or Secp256k1 - DID Document published to index (verification method) - Readers derive public key from xpub or verify binding proof

Verification Process:
```
1. Fetch record from gateway 2. Extract signature and DID reference 3. Resolve DID Document from index 4. Derive public key (xpub mode)
```

Acceptance Criteria: - Publish signed record - Reader verifies signature independently - Invalid signature detected and flagged - Revoked keys marked as invalid

Feature 10: No Monthly Fees

Description: Self-hosted on home network with no ongoing costs.

User Benefit: Freedom of speech shouldn't require subscription fees or cloud hosting.

Technical Implementation: - All software open source - Runs on consumer hardware (Raspberry Pi, NUC, old laptop) - Peer-to-peer distribution spreads bandwidth costs - Optional donations to gateway operators (future)

Cost Breakdown: - **Hardware**: $35 (Raspberry Pi 4) - $200 (NUC) one-time - **Network**: Existing home internet (no additional cost) - **Blockchain Fees**: Optional (someone else can sponsor) - **Ongoing**: $0/month

Acceptance Criteria: - Deploy stack on Raspberry Pi without paid services - Publish records without blockchain fee requirement (sponsored) - Retrieve content via peer networks (no bandwidth bills) - System runs indefinitely without subscription

Feature 11: Decentralized Name Resolution

Description: Dual-path name resolution system using ENS (Ethereum Name Service) for mainstream reach and GNS (GNU Name System) for private lookups.

User Benefit: Human-readable names without central DNS authorities. Choose between fast mainstream resolution (ENS) or private lookups that don't expose queries (GNS).

Technical Implementation:

Canonical Service Descriptor

Each gateway publishes a standardized service descriptor containing:
`json { "version": 1, "https": ["https://gateway.example.org"], "onion": ["http://abcd1234.onion"], "ipfs": ["ipfs://bafy..."], "api": [`

Dual Publishing System

ENS Path (mainstream, fast): - Update `contenthash` to point to IPFS/IPNS index - Store descriptor in `TXT` record (minified ≤ 1.5 KB) - Optional `SRV` records for non-HTTP services - Signed with JWS for integrity verification - Resolution via standard ENS resolvers and gateways (e.g., `.limo`)

GNS Path (private, no query leakage): - Equivalent zone records in GNUnet - Same descriptor distributed across GNS records - Short TTLs for endpoint rotation - Signed zone data for integrity - Resolution only via local GNUnet daemon

Name Resolution Modes

Fast Mode (default): - Query ENS first via public resolvers - P95 resolution ≤ 300 ms - Works in standard browsers via gateway services - Query metadata potentially visible to resolver operators

Private Mode (opt-in): - Query GNS only via local GNUnet daemon - No external query visibility - P95 resolution ≤ 800 ms (with local GNUnet stack) - Requires GNUnet installation - Optional fallback to ENS if configured

Resolver Helper Service

Local service running on `127.0.0.1:4777`: - Exposes HTTP API: `GET /v1/resolve?name=<n>&mode=fast|private` - Verifies JWS signatures on retrieved descriptors - Caches resolved descriptors with TTL respect - Returns structured JSON for application consumption

Response Format:
`json { "name": "gateway.eth", "source": "ens", "descriptor": { "https": ["https://gateway.example.org"], "onion": ["http://abcd1234.oni`

Browser Extension (Optional)

- Intercepts configured names (`.eth`, `.gnu`)
- Fetches resolution from local helper service
- Shows mode badge: **FAST** or **PRIVATE**
- Redirects to resolved endpoint automatically

Operator Workflow

1. **Define Service**: Create canonical descriptor YAML/JSON
2. **Publish to ENS**: Update contenthash + TXT records via configured wallet
3. **Publish to GNS**: Update zone records via GNUnet
4. **Verify**: Automated read-back verification for both systems
5. **Audit**: All publishes logged with transaction IDs

Name Publisher Utility: ```bash

Publish to both ENS and GNS

./publish-name.sh --descriptor service.json --ens-name gateway.eth --gns-zone gateway.gnu

Verify propagation

./verify-name.sh gateway.eth # Check ENS ./verify-name.sh gateway.gnu --mode private # Check GNS ```

Security and Privacy

Requester Privacy: - **ENS**: Queries may be visible to resolver operators and DNS providers - **GNS**: All queries resolved locally, no external visibility - Helper warns users when in fast mode about query visibility

Host Anonymity: - Name resolution alone does not provide host anonymity - Pair with Tor onion addresses in descriptor for origin privacy - `.onion` addresses included in descriptor for anonymous access

Integrity: - All descriptors signed with detached JWS - Public key pinned in resolver configuration - Clients verify signature before accepting descriptor - Prevents man-in-the-middle attacks and stale gateway hijacking

Key Management: - **ENS**: Hardware wallet support (Ledger, Trezor) or custodial signer - **GNS**: Zone keys in encrypted keystore with offline backup procedure - No private keys stored on servers by default

Endpoint Rotation

Operators can rotate endpoints transparently: 1. Update canonical descriptor with new endpoints 2. Publish to both ENS and GNS atomically 3. Old descriptors expire based on TTL 4. Clients honor `updated` timestamp and `ttl` hint 5. Gradual migration as clients refresh cache

ICANN DNS Fallback

For maximum compatibility, operators can also configure: - Traditional DNS `A` record → clearnet gateway IP - `TXT` record with same signed descriptor - Lowest security but works everywhere - Documented in setup guide as "mainstream fallback"

Acceptance Criteria: - Single descriptor publishes to both ENS and GNS - Read-back from both systems matches original descriptor - Helper resolves ENS-only name in fast mode (<300ms P95) - Helper resolves GNS-only name in private mode (<800ms P95) - JWS verification fails on tampered descriptors - Switching modes updates resolution path without restart - 95%+ successful resolves in fast mode - 90%+ successful resolves in private mode (with GNUnet installed) - <1% signature mismatches after propagation

---

Technical Specifications

Identity & Authentication (OIP 0.9)

DID Method Specification

DID Format: `did:key:z6Mk...` (master identity)
DID Documents: Stored as OIP records, referenced by `did:arweave:txid`

Components:
`json { "didDocument": { "did": "did:key:z6Mk...", "controller": "did:key:z6Mk...", "verificationMethod": [ "did:arweave:vm_xpub_tx", "d`

Verification Method Types

Type 1: xpub-based (non-hardened leaf)
`json { "vm_type": "oip:XpubDerivation2025", "xpub": "xpub6CUGRUonZSQ...", "derivation_sub_purpose": "identity.sign", "derivation_accoun`

Type 2: Binding proof (hardened leaf)
`json { "vm_type": "Ed25519VerificationKey2020", "publicKeyMultibase": "z6MkfY...child", "derivation_sub_purpose": "identity.sign", "lea`

Signature Algorithm

Signing Process: 1. Assemble `DataForSignature` object (all record fields except signature) 2. Canonicalize to deterministic JSON (sorted keys, LF line endings) 3. Hash with SHA256 4. Sign hash with child private key (Ed25519 or Secp256k1) 5. Base64url encode signature 6. Attach as `CreatorSig` tag or field

Verification Process: 1. Parse DID from record 2. Resolve DID Document 3. Select appropriate verification method 4. Derive public key (xpub mode) or verify binding proof (hardened mode) 5. Recompute canonical hash 6. Verify signature with derived public key

Publisher Gateway API

Endpoint: POST /publish

Description: Accept signed record submission via Tor

Request: ```http POST /publish HTTP/1.1 Host: .onion Content-Type: application/json Authorization: Bearer Idempotency-Key:

{ "record": { "basic": { "name": "Article Title", "description": "Summary", "date": 1736985600, "tagItems": ["whistleblower", "corruption"] }, "post": { "articleText": "Full article content...", "bylineWriter": "Anonymous Journalist" } }, "signature": { "did": "did:key:z6Mk...", "verificationMethod": "#sign", "jws": "eyJhbGciOiJFZERTQSJ9..." }, "meta": { "wordpress": { "post*id": 42, "post*type": "post" } } } ```

Response: ```http HTTP/1.1 202 Accepted Content-Type: application/json

{ "submissionId": "sub_abc123", "status": "queued", "estimatedStart": "2025-01-15T12:34:56Z" } ```

Error Codes: - `400` - Invalid record schema - `401` - Invalid signature or token - `413` - Record exceeds size limit - `415` - Unsupported MIME type in media - `429` - Rate limit exceeded - `503` - Gateway temporarily unavailable

Endpoint: GET /status/:submissionId

Description: Check submission processing status

Response:
`json { "submissionId": "sub_abc123", "status": "succeeded", "outputs": { "did": "did:arweave:new_tx_id", "arweave_txid": "new_tx_id", "`

Status Values: - `queued` - Awaiting processing - `processing` - Being validated and stored - `succeeded` - Published and indexed - `failed` - Rejected (see `error` field)

Endpoint: GET /queue/limits

Description: Get gateway moderation policies

Response:
`json { "maxBodyMb": 25, "maxAttachments": 10, "mimeAllow": ["image/jpeg", "image/png", "video/mp4", "application/pdf"], "tagPolicy": {`

Media Distribution

Media Manifest Structure

Included in Record Metadata:
`json { "media": [ { "filename": "document.pdf", "size": 2458624, "sha256": "abc123...", "mimeType": "application/pdf", "distribution":`

Media Upload Flow

1. **WordPress**: Author attaches file to post
2. **Plugin**: Generates SHA256 hash (mediaId)
3. **Seeder**: Creates BitTorrent torrent, seeds via WebTorrent
4. **Plugin**: Generates manifest with magnet URI
5. **Gateway**: Optionally pins to IPFS, adds HTTP mirror
6. **Record**: Published with media manifest (not file contents)
7. **Readers**: Retrieve via BitTorrent/IPFS/HTTP (peer network)

Key Principle: Large files never go over Tor—only small metadata manifests.

Gateway Moderation Policies

Abuse-Aware Controls

Size Caps: - Default: 25MB for record metadata (not media files) - Configurable per gateway - Prevents storage exhaustion attacks

MIME Type Allowlists: - Gateway defines permitted content types - Example: Allow documents, images, videos; block executables - Prevents malware distribution

Hash Denylists: - Known CSAM/malware hashes blocked - Industry-standard lists (NCMEC, VirusTotal) - Content never stored or served

Rate Limits: - Per-publisher DID throttling - Example: 10 submissions/hour/DID - Prevents spam and DoS

Topic-Scoped Queues: - Tag-based auto-approval - Example: "research" → auto-approve; "sensitive" → review queue - Allows trusted publishers to bypass review

Moderation Actions

Approve: Index and serve record publicly
Quarantine: Index but mark for review; show in admin panel only
Hide: Index but don't serve; preserve for legal/transparency
Reject: Don't index; return error code to publisher

Key Principle: Decisions are per-gateway. Content rejected by one gateway may be approved by another.

Search and Discovery

Elasticsearch Index Structure

Record Document: `json { "oip": { "did": "did:arweave:tx_id", "recordType": "post", "storage": "arweave", "indexedAt": "2025-01-15T12:35:30Z", "creator":`

Query Capabilities

Full-Text Search: - Search across all text fields - Weighted by relevance (title > description > content) - Tag-based filtering

Field-Specific Queries: - By author DID - By date range - By record type - By media type (image, video, document)

Alfred RAG Integration: - Natural language queries - Context-aware responses - Citation with DIDs - Local processing (no cloud)

Alfred (Local AI/RAG)

Architecture

Components: - **Ollama**: Local LLM server (llama3.2:3b default) - **Elasticsearch**: Vector search + full-text - **RAG Pipeline**: Question analysis → search → context building → generation - **Voice Interface**: Speech-to-text (Whisper) + text-to-speech (Kokoro/ElevenLabs)

Capabilities

Question Understanding: - "What records mention XYZ company?" → Search + summarize - "Who published research on topic ABC?" → Author identification - "Find documents signed by DID xyz123" → Provenance search

Context Building: - Fetch relevant records from Elasticsearch - Extract full text from referenced content - Build context window for LLM - Cite sources with DIDs

Response Generation: - Local LLM processes query + context - Generates answer with citations - Returns sources for verification - No data sent to external services

Privacy: - All processing on-device - No cloud API calls - Conversation history encrypted locally - Voice data never leaves machine

---

Implementation Phases

Current Status: OIP 0.8 Foundation (COMPLETE ✅)

Already Implemented:

- [x] Docker Compose deployment (full stack)
- [x] OIP spec v0.8 with template-record system
- [x] Arweave integration for public permanent records
- [x] GUN integration for private encrypted records
- [x] HD wallet authentication system (BIP-39, BIP-32)
- [x] Abuse-aware controls (size caps, MIME filters, rate limits)
- [x] BitTorrent/WebTorrent media distribution
- [x] IPFS integration
- [x] MediaSeeder service for persistent seeding
- [x] Alfred RAG system with local AI (Ollama)
- [x] Advanced search with Elasticsearch
- [x] Reference client web interface
- [x] Mac client with voice interface
- [x] Cross-gateway GUN synchronization
- [x] User registration and authentication (username + password)
- [x] Media streaming with range request support
- [x] Organization access control
- [x] Multi-LLM support (OpenAI, XAI, Ollama)
- [x] Conversation history with encryption

Status: Production-ready foundation is complete. WeAreChange.org and TimCast.com are currently using OIP 0.8.

Phase 1: OIP 0.9 Identity System (Weeks 1-3, ~21 days)

Deliverables:

- [ ] DID-based identity system (OIP 0.9 spec)
- [ ] HD wallet key derivation with SLIP-0043 paths
- [ ] `didDocument` and `didVerificationMethod` templates
- [ ] Signature verification (xpub mode + binding proofs)
- [ ] Sub-purpose key derivation (sign, encrypt, delegate, revoke, etc.)
- [ ] Rolling key policy and revocation support
- [ ] Create-but-don't-send functionality (.lopkg package format)

Success Criteria:

- Generate DID from HD wallet master key
- Sign records with purpose-scoped child keys
- Verify signatures via xpub derivation or binding proofs
- Export signed packages offline
- Import and submit packages from different network

Technical Work:

- Implement DID Document template and resolution
- Add SLIP-0043 key derivation (`m/43'/176800'/sub-purpose'/account'/index`)
- Build signature verification with multiple paths
- Create `.lopkg` package format specification
- Update API endpoints to support DID signatures
- Migrate existing HD wallet system to OIP 0.9 paths
- Add key rotation and revocation mechanisms

Timeline: 3 weeks (parallel development possible)

Phase 2: WordPress Plugin & Tor Integration (Weeks 4-8, ~35 days)

Deliverables:

- [ ] WordPress plugin ("LO Publisher" / "DWeb Publisher")
- [ ] Gutenberg editor integration
- [ ] Local DID key management in WordPress
- [ ] Field mapping UI (WP fields → OIP templates)
- [ ] Media manifest generation (no large files over Tor)
- [ ] Tor onion service for publisher gateway
- [ ] Publisher Gateway API (loopback-only)
- [ ] Gateway moderation queue UI
- [ ] Batched submission with timing obfuscation
- [ ] Header stripping and privacy hardening

Success Criteria:

- Install WordPress + plugin → compose post → publish to OIP via Tor
- Export signed package from WordPress
- Submit via `.onion` address without revealing IP
- Gateway moderates submissions before indexing
- Timing analysis shows no one-to-one correlation

Technical Work:

- WordPress plugin development (PHP + JS)
- DID key storage in WordPress (encrypted)
- Tor daemon integration and configuration
- Publisher Gateway service (Node.js/Express)
- Moderation queue backend and UI
- Idempotency and status tracking
- `/publish` , `/status/:id` , `/queue/limits` endpoints
- Integration tests for end-to-end flow

Timeline: 5 weeks

Phase 3: Decentralized Name Resolution (Weeks 9-13, ~35 days)

Deliverables:

- [ ] Canonical service descriptor format
- [ ] ENS publisher utility (contenthash + TXT records)
- [ ] GNS publisher utility (GNUnet zone records)
- [ ] JWS signing for descriptor integrity
- [ ] Resolver helper service (fast + private modes)
- [ ] Browser extension (optional)
- [ ] Name publisher CLI tool
- [ ] Verification and audit logging

Success Criteria:

- Single descriptor publishes to ENS and GNS
- Helper resolves ENS name in fast mode (<300ms P95)
- Helper resolves GNS name in private mode (<800ms P95)
- JWS verification fails on tampered descriptors
- Mode switching works without restart
- 95%+ success rate for fast mode
- 90%+ success rate for private mode (with GNUnet)

Technical Work:

- Service descriptor schema definition
- ENS integration with Web3 providers
- GNUnet integration for GNS publishing
- JWS signing and verification library
- Resolver helper HTTP API (port 4777)
- Browser extension development (Chrome/Firefox)
- Configuration management for keys
- Endpoint rotation handling
- Integration with OIP gateway setup

Timeline: 5 weeks

Phase 4: Raspberry Pi Image & Deployment Tools (Weeks 14-18, ~35 days)

Deliverables:

- [ ] Raspberry Pi OS image with full stack
- [ ] First-run setup wizard (web UI)
- [ ] Automatic service configuration
- [ ] DID key generation wizard
- [ ] Gateway moderation policy setup
- [ ] Health monitoring dashboard
- [ ] Automatic updates mechanism
- [ ] Backup and restore tools

Success Criteria:

- Flash SD card → boot → complete setup in <10 minutes
- Non-technical user can deploy successfully
- System runs 30+ days without intervention
- Setup wizard configures all services correctly
- Automatic updates work seamlessly

Technical Work:

- Raspberry Pi OS customization
- Service orchestration and health checks
- Web-based setup wizard (React/Vue)
- Configuration management system

- Update mechanism with rollback support
- System monitoring and alerting
- Documentation and video tutorials
- Testing on Pi 4 and Pi 5 hardware

Timeline: 5 weeks

Phase 5: Ecosystem Growth & Community (Weeks 19+, Ongoing)

Deliverables:

- [ ] Kiosk mode for shared deployments
- [ ] Archive.org backup integration
- [ ] Mobile apps (iOS/Android)
- [ ] Desktop apps (macOS/Windows/Linux)
- [ ] Gateway discovery service
- [ ] Operator community forum
- [ ] Case studies and success stories
- [ ] Marketing and outreach materials

Success Criteria:

- 10+ production gateways by end of Phase 4
- 100+ gateways by 6 months post-launch
- 1000+ publishers by end of year 1
- 10,000+ records indexed across network
- Active community of operators and developers

Technical Work:

- Kiosk mode isolation and multi-user support
- Archive.org API integration for automatic backups
- Native mobile app development
- Native desktop app development
- Gateway registry and discovery service
- Community infrastructure (forums, chat, wiki)
- Documentation site and tutorials
- Conference presentations and workshops
- Partnership development with journalism organizations

Timeline: Ongoing after initial release

Summary Timeline

Phase	Duration	Cumulative	Status
Foundation (0.8)	Complete	-	✅ Production
Phase 1: OIP 0.9	3 weeks	3 weeks	🔄 Next
Phase 2: WordPress + Tor	5 weeks	8 weeks	⏳ Pending
Phase 3: Name Resolution	5 weeks	13 weeks	⏳ Pending
Phase 4: Pi Image	5 weeks	18 weeks	⏳ Pending
Phase 5: Ecosystem	Ongoing	-	⏳ Pending

Total to MVP: 18 weeks (~4.5 months) from start of Phase 1

Key Dependencies: - Phase 2 depends on Phase 1 (DID system required for signing) - Phase 3 can run parallel to Phase 2 (independent systems) - Phase 4 depends on Phases 1-3 (requires all components working) - Phase 5 begins during Phase 4 (parallel community building)

Success Metrics

Publisher Metrics

Anonymity: - 0% of submissions contain publisher IP addresses - 0% timing correlation between Tor submission and clearnet publish - 100% of .onion submissions succeed without identity leakage

Usability: - <10 minutes to complete setup (from image to first publish) - <2 seconds from "Publish" click to submission receipt - >90% of submissions succeed on first attempt - <5% rejections due to policy violations (after initial learning)

Adoption: - 100+ active publishers in first 6 months - 1000+ active publishers by end of year 1 - 50% of publishers use WordPress plugin - 30% of publishers use create-but-don't-send

Gateway Metrics

Reliability: - >99.5% uptime for gateway API - <60 seconds to index approved submissions - <5 minutes for cross-gateway synchronization - >95% of media available via peer networks

Moderation: - <1% false positive rejections (valid content blocked) - <0.1% false negatives (abusive content approved) - <24 hours for manual review queue items - >80% of submissions auto-approved (after policy tuning)

Discoverability: - >3 gateways index each record (redundancy) - <5 seconds for full-text search queries - >90% of searches return relevant results - 100% of records verifiable via DID signatures

Reader Metrics

Discovery: - >50% of queries satisfied by first page of results - <3 seconds for search result rendering - >80% of records have available content (via peer networks) - <10 seconds to verify record signature

Alfred RAG: - >80% of questions answered correctly - <10 seconds for RAG query response - 100% of processing happens locally (no cloud calls) - >70% user satisfaction with voice interface

Ecosystem Metrics

Resilience: - Records remain discoverable when 50% of gateways offline - Content retrievable when original publisher offline - Zero single points of failure in architecture - <1 hour to deploy new gateway node

Cost: - $0/month ongoing costs for home deployments - <$100 one-time hardware cost (Raspberry Pi setup) - <$500 one-time cost for higher-end gateway (NUC/server) - Optional blockchain fees sponsored by others

Adoption: - 10 production gateways by month 6 - 100 production gateways by end of year 1 - 5 organizations (journalism/activism) using as primary platform - 50% of gateways operated from home networks

---

Risks & Mitigations

Risk 1: Timing Correlation Attacks

Description: Adversary observes Tor submission timing and correlates with blockchain publish time to identify publisher.

Impact: HIGH - Deanonymizes publishers despite Tor

Likelihood: MEDIUM - Requires global network view

Mitigation: - Batch submissions with randomized delays (60s ± 30s jitter) - Mix records from multiple publishers in each batch - Vary delay windows based on network load - Document timing attack surface for operators

Residual Risk: LOW - Requires sustained observation of both Tor and blockchain

Risk 2: Gateway Operator Liability

Description: Operator faces legal action for content indexed by their gateway.

Impact: MEDIUM - Operators shut down nodes due to legal pressure

Likelihood: MEDIUM - Depends on jurisdiction and content

Mitigation: - Gateway-scoped moderation gives operators control - Abuse-aware controls (MIME filters, hash denylists) - Clear documentation of legal safe harbors (DMCA, Section 230) - Community legal defense fund (future) - Operators can run gateway anonymously via Tor

Residual Risk: MEDIUM - Legal landscape varies by jurisdiction

Risk 3: Storage Exhaustion Attacks

Description: Attacker floods gateway with large records to exhaust storage.

Impact: MEDIUM - Gateway becomes unavailable

Likelihood: MEDIUM - Easy to execute without mitigation

Mitigation: - Size caps on record metadata (25MB default) - Rate limits per publisher DID (10 submissions/hour) - Storage quotas per gateway (configurable) - Media files distributed via peer networks (not stored on gateway)

Residual Risk: LOW - Controls make attack economically infeasible

Risk 4: Sybil Attacks (Spam Publishers)

Description: Attacker creates many DID identities to evade rate limits.

Impact: MEDIUM - Gateway spam reduces signal-to-noise

Likelihood: HIGH - DIDs are free to generate

Mitigation: - Proof-of-work per submission (optional, future) - Reputation systems for DID identities (future) - Tag-based auto-approval (trusted publishers bypass review) - Community curation of high-quality gateways

Residual Risk: MEDIUM - Arms race between spammers and defenders

Risk 5: Content Moderation Challenges

Description: Balance between censorship resistance and preventing abusive content.

Impact: HIGH - Platform reputation and operator liability

Likelihood: HIGH - Will encounter abusive content

Mitigation: - Gateway-scoped moderation (no global decisions) - Industry-standard hash denylists (CSAM, malware) - Transparent moderation policies per gateway - Multiple gateways with different policies - Legal safe harbor protections for operators

Residual Risk: MEDIUM - Ongoing challenge requiring community norms

Risk 6: Key Loss or Compromise

Description: Publisher loses mnemonic or private key is stolen.

Impact: HIGH for affected publisher - Loses identity and cannot publish

Likelihood: MEDIUM - User error or device compromise

Mitigation: - 12-word mnemonic backup during setup - Encrypted mnemonic storage with password - Key rotation supported (rolling keys) - Revocation mechanism via DID Documents - Hardware key support (future)

Residual Risk: MEDIUM - User responsibility for key management

Risk 7: Blockchain Backend Failure

Description: Arweave network becomes unavailable or prohibitively expensive.

Impact: HIGH - Cannot persist new records

Likelihood: LOW - Arweave has strong track record

Mitigation: - DID spec means metadata can move to different chain - Already abstracted at application layer - Can switch to Bitcoin, Ethereum, or other backend - Local DHT mode for temporary outages (future)

Residual Risk: LOW - System is blockchain-agnostic

Risk 8: Tor Network Attacks

Description: Tor network compromised or blocked by nation-states.

Impact: HIGH - Publishers cannot submit anonymously

Likelihood: MEDIUM - Ongoing threat from state actors

Mitigation: - Create-but-don't-send allows offline relay - I2P integration as Tor alternative (future) - VPN + Tor for defense in depth - Pluggable transports to evade blocking

Residual Risk: MEDIUM - Nation-state adversaries are persistent

Risk 9: Low Adoption / Network Effects

Description: Insufficient publishers and gateways to achieve critical mass.

Impact: HIGH - Platform fails to gain traction

Likelihood: MEDIUM - Many decentralized platforms struggle with adoption

Mitigation: - Partnership with journalism organizations (FPF, EFF) - Integration with existing WordPress sites (large user base) - One-click deployment lowers barrier to entry - Alfred RAG provides unique value (local AI) - Historical OIP credibility (DLOA, Tim Berners-Lee endorsement)

Residual Risk: MEDIUM - Market risk inherent to new platforms

Risk 10: Regulatory Capture

Description: Governments mandate backdoors, logging, or identification requirements.

Impact: HIGH - Defeats core anonymity promise

Likelihood: LOW to MEDIUM - Depends on jurisdiction

Mitigation: - Open source → forks cannot be prevented - Distributed development (no single legal entity) - Operators can run nodes in friendly jurisdictions - Technical design makes backdoors infeasible (no central control)

Residual Risk: MEDIUM - Varies by operator location

Appendix A: Comparison to Alternatives

DWeb Server vs. Traditional Blogging Platforms

Feature	DWeb Server	Medium/Substack	WordPress.com
Anonymity	Tor + DID (no registration if self-hosted; username + password if using gateway)	Email required	Email required
Censorship Resistance	Distributed index	Central platform	Central platform
Cost	$0/month	$0-$50/month	$0-$25/month
Discovery	Cross-gateway search	Platform discovery	Platform SEO
Control	Full self-hosting	Platform TOS	Platform TOS
Takedowns	Gateway-scoped only	Platform-wide	Platform-wide

DWeb Server vs. Decentralized Social Media

Feature	DWeb Server	Mastodon	Bluesky
Anonymity	Tor + DID	Email required	Email required
Persistence	Blockchain + peers	Server-dependent	Platform-dependent
Moderation	Gateway-scoped	Server-scoped	Platform-scoped
Setup	One-click Pi image	Server setup required	Account registration
Long-Form	Full articles	Short posts	Short posts
Media	P2P distribution	Server storage	Platform storage

DWeb Server vs. Existing DWeb Projects

Feature	DWeb Server	ZeroNet	IPFS + ENS
Ease of Use	One-click deploy	Complex setup	Very complex
Authoring	WordPress	HTML editing	Manual
Anonymity	Tor built-in	Tor optional	No Tor
Discovery	Indexed gateways	DHT only	Hard to find content
Moderation	Gateway-scoped	None	None
Active Development	✅	Inactive	Active

DWeb Server vs. SecureDrop

Feature	DWeb Server	SecureDrop
Use Case	Anonymous publishing	Whistleblower submissions
Audience	Public readership	Specific journalists
Persistence	Permanent index	Temporary submission
Discovery	Full-text search	Not applicable
Setup	One-click Pi image	Requires IT staff
Cost	$35 (Pi)	$500+ (servers)

Key Differentiator: DWeb Server combines the **ease of WordPress** with the **anonymity of Tor**, the **permanence of blockchain**, and the **resilience of peer networks**—a unique combination not found in existing platforms.

Appendix B: User Onboarding Flow

First-Time Publisher Setup

Step 1: Install (5 minutes)

- Option A: Download Raspberry Pi image, flash SD card
- Option B: `git clone` + `docker-compose up`
- Option C: Download desktop app (Mac/Windows/Linux)

Step 2: Setup Wizard (3 minutes)

- Choose deployment mode: Publisher + Gateway, or Publisher-only
- Generate DID identity (12-word mnemonic displayed)
- **CRITICAL**: User writes down mnemonic backup
- Set gateway moderation policies (if running gateway)

Step 3: Configure WordPress (2 minutes)

- Install LO Publisher plugin (one-click from dashboard)
- Import DID key from setup wizard
- Configure target gateway(s): `.onion` address + token
- Select submission mode: Direct or Export-only

Step 4: Publish First Record (5 minutes)

- Write test post in WordPress
- Click "Publish to OIP" button
- Review submission preview (shows what will be signed)
- Confirm → record signed and submitted
- Receive submission receipt with ID

Step 5: Verify Publication (2 minutes)

- Check status via plugin dashboard
- Wait for gateway to index (~60 seconds)
- Search for record on gateway web UI
- Verify signature shows your DID
- Content now discoverable by others

Total Time: 17 minutes from zero to first publication

Subsequent Publications

- Write in WordPress as normal
- Click "Publish to OIP"
- Receive instant confirmation
- Content indexed within 60 seconds

Time per Post: <30 seconds (after first setup)

Appendix C: Gateway Operator Guide

Running a Gateway Node

Hardware Requirements (Minimum):

- Raspberry Pi 4 (4GB RAM)
- 128GB SD card or SSD
- Stable internet connection (10 Mbps+)

Hardware Requirements (Recommended):

- Intel NUC or similar (8GB+ RAM)
- 500GB SSD
- 50 Mbps+ internet
- UPS for power reliability

Software Requirements:

- Linux (Ubuntu 22.04 or Raspberry Pi OS)
- Docker + Docker Compose
- Tor daemon (v3 onion service)

Services Included:

- Gateway API (onion + clearnet)
- Elasticsearch (index layer)
- GUN relay (synchronization)
- MediaSeeder (BitTorrent/IPFS)
- Alfred (local AI/RAG)

Installation:

```
# Clone repository
git clone https://github.com/oip/lapis-obscura.git
cd lapis-obscura

# Configure environment
cp example.env .env
nano .env  # Set COMPOSE_PROJECT_NAME, moderation policies

# Start stack
docker-compose up -d

# Access setup wizard
open http://localhost:3005/setup
```

Moderation Policy Configuration:

```
# .env file
GATEWAY_MAX_BODY_MB=25
GATEWAY_MIME_ALLOWLIST=image/jpeg,image/png,video/mp4,application/pdf
GATEWAY_AUTO_APPROVE_TAGS=research,journalism
GATEWAY_REVIEW_QUEUE_TAGS=sensitive
GATEWAY_RATE_LIMIT_RPS=5
GATEWAY_HASH_DENYLIST_URL=https://example.com/denylist.txt
```

Operating Costs:

- Electricity: ~$2-5/month (Raspberry Pi)
- Internet: $0 additional (uses existing connection)
- Hardware: $35-200 one-time
- **Total**: ~$3/month average

Legal Considerations:

- Run gateway anonymously via Tor (optional)
- Use VPS in privacy-friendly jurisdiction (optional)
- Implement strong moderation policies
- Document policies publicly
- Consider Section 230 / DMCA safe harbor (US)
- Consult local counsel for jurisdiction-specific advice

Appendix D: Frequently Asked Questions

For Publishers

Q: Can my identity be revealed?
A: Not through network analysis. Tor hides your IP, DID keys are pseudonymous. If you run your own node, no registration is required at all. If you use an existing gateway, registration is just username + password (no email verification). However, content itself may reveal identity (writing style, metadata), so operational security is your responsibility.

Q: Do I need to register an account?
A: **If you run your own DWeb Server node**: No registration required. Generate your DID from a mnemonic and publish immediately. Full self-sovereignty.

If you use someone else's gateway: Yes, simple registration (username + password only, no email). Your encrypted mnemonic is stored on the gateway server so you can publish from any device. You can export your mnemonic and migrate to self-hosting anytime.

Trade-off: Self-hosting = maximum control but requires technical setup. Gateway registration = convenience but trusted server stores your encrypted keys.

Q: What if I lose my 12-word mnemonic?
A: You lose access to your DID identity permanently. Always back up your mnemonic securely (write it down, store in safe place). Cannot be recovered.

Q: Can my content be taken down?
A: Individual gateways can choose not to index your content, but it remains on the blockchain and other gateways may still serve it. No single entity can remove it globally.

Q: How much does it cost to publish?
A: $0 for you. Blockchain fees (if any) can be sponsored by gateway operators or others. No monthly hosting fees.

Q: Can I edit or delete published content?
A: Records are immutable once published. You can publish updated versions and mark old versions as "superseded" via your DID. Deletions are not possible (by design).

For Gateway Operators

Q: Am I liable for content on my gateway?
A: Varies by jurisdiction. You have moderation controls and can refuse to index/serve content. Consult legal counsel. Many jurisdictions have safe harbor protections for platforms.

Q: Can I run a gateway from home?
A: Yes! Designed for home network deployment. Use dynamic DNS or Tor onion address. Bandwidth requirements are modest (<10 Mbps for small gateway).

Q: How much storage is needed?
A: Elasticsearch index grows slowly (~1GB per 10,000 records). Media files can be stored on gateway or only seeded via BitTorrent (operator choice).

Q: Can I shut down my gateway?
A: Yes. Content persists on blockchain and other gateways. Shutting down your gateway only affects its index, not the broader network.

For Readers

Q: How do I find content?
A: Search any gateway's web interface, or use Alfred (local AI) to query your preferred gateway. Content discoverable across all gateways.

Q: How do I verify content authenticity?
A: Each record has a DID signature. Gateway web UI shows verification status. You can independently verify by deriving public key from DID Document.

Q: What if a gateway is down?
A: Try a different gateway. Records replicated across multiple gateways. Content seeded via peer networks remains available.

Q: Can I download content for offline reading?
A: Yes. Use BitTorrent magnet URIs to download files. Records can be exported as JSON for archival.

Technical

Q: Why not use IPFS for everything?
A: IPFS is great for content, but metadata discoverability is hard. Blockchain + Elasticsearch provides queryable index. We use IPFS for optional content distribution.

Q: Why Arweave and not Bitcoin?
A: Arweave designed for permanent data storage with one-time fee. Bitcoin OP_RETURN is limited to 80 bytes. But DID spec means we can switch to Bitcoin if needed.

Q: What happens if Arweave disappears?
A: DID spec abstraction means we can migrate to different blockchain. Application layer unchanged. Gateway operators can switch persistence backend.

Q: Why WordPress and not Ghost/Hugo/other?
A: WordPress is most widely known and used. We may add Ghost/Hugo plugins later. WordPress gives us largest potential user base.

Appendix E: Roadmap to 1.0 Release

Version 0.1 (Alpha) - Week 3 (End of Phase 1)

Timeline: 3 weeks from kickoff

Features:

- [x] Docker Compose deployment (OIP 0.8 foundation - already complete)
- [x] Gateway moderation policies (OIP 0.8 - already complete)
- [x] Cross-gateway synchronization (OIP 0.8 - already complete)
- [x] BitTorrent media distribution (OIP 0.8 - already complete)
- [x] Alfred RAG integration (OIP 0.8 - already complete)
- [x] IPFS integration (OIP 0.8 - already complete)
- [x] Advanced search with Elasticsearch (OIP 0.8 - already complete)
- [ ] DID-based identity system (OIP 0.9)
- [ ] HD wallet with SLIP-0043 key derivation
- [ ] `didDocument` and `didVerificationMethod` templates
- [ ] Signature verification (xpub + binding proofs)
- [ ] Create-but-don't-send (.lopkg package format)
- [ ] Multi-signature publishing
- [ ] Delegation credentials

Status: Foundation (0.8) complete; DID system (0.9) in development

Users: Internal testing and early adopters

Key Milestone: OIP 0.9 spec complete with full DID-based identity

---

Version 0.5 (Beta) - Week 8 (End of Phase 2)

Timeline: 5 weeks after 0.1 Alpha

Features:

- [ ] WordPress plugin ("DWeb Publisher")
- [ ] Gutenberg editor integration
- [ ] Local DID key management in WordPress
- [ ] Field mapping UI (WP fields → OIP templates)
- [ ] Media manifest generation
- [ ] Tor onion service for publisher gateway
- [ ] Publisher Gateway API (loopback-only)
- [ ] Gateway moderation queue UI
- [ ] Batched submission with timing obfuscation
- [ ] Header stripping and privacy hardening
- [ ] Name resolution (ENS + GNS dual-path)
- [ ] Resolver helper service (fast + private modes)
- [ ] Browser extension for name resolution
- [ ] Canonical service descriptor format

Status: WordPress + Tor + Name Resolution integration

Users: Friendly journalists and activists (10-50)

Key Milestones: - Anonymous publishing via Tor working end-to-end - Human-readable names via ENS/GNS - WordPress as primary authoring tool

---

Version 0.8 (Release Candidate) - Week 13 (End of Phase 3)

Timeline: 5 weeks after 0.5 Beta (can run parallel with Phase 2)

Features:

- [ ] Raspberry Pi image (complete stack)
- [ ] First-run setup wizard (web UI)
- [ ] Automatic service configuration
- [ ] DID key generation wizard
- [ ] Gateway moderation policy setup UI
- [ ] Health monitoring dashboard
- [ ] Automatic updates mechanism
- [ ] Backup and restore tools
- [ ] Archive.org backup integration
- [ ] Automated clearnet snapshots for archival
- [ ] Integration with Internet Archive APIs

Status: Production-ready deployment with one-click setup

Users: Open beta (100-500)

Key Milestones: - Non-technical users can deploy from Raspberry Pi image - Automatic backups to archive.org working - 10+ production gateway nodes operational

---

Version 1.0 (Production) - Week 18 (End of Phase 4)

Timeline: 5 weeks after 0.8 RC

Features:

- [x] All 0.1-0.8 features stable and tested
- [ ] Security audit complete (external firm)
- [ ] Penetration testing complete
- [ ] Performance optimization
- [ ] Production documentation (operator guides, user manuals)
- [ ] Video tutorials and onboarding materials
- [ ] Community forum and support infrastructure
- [ ] Incident response procedures
- [ ] Legal compliance documentation
- [ ] Brand and marketing materials

Status: Public release ready

Users: General public

Success Criteria:

- 100+ active gateway nodes
- 1,000+ publishers
- 10,000+ records indexed
- <0.1% critical bugs
- P95 publish latency <5 seconds
- P95 search latency <1 second
- Security audit passed with no critical findings

Key Milestone: DWeb Server 1.0 production launch

Post-1.0 Enhancements

Version 1.5 (6 months post-1.0):

- [ ] Mobile apps (iOS/Android)
- [ ] Desktop apps (native - macOS/Windows/Linux)
- [ ] Kiosk mode for shared deployments
- [ ] VNC-style "reporting desk" mode
- [ ] Enhanced monitoring and alerting
- [ ] Performance dashboards for operators
- [ ] Community reputation systems

Version 2.0 (12 months post-1.0):

- [ ] Hardware key support (YubiKey, Ledger)
- [ ] Advanced key management features
- [ ] Enhanced reputation systems
- [ ] Tipping/subscriptions (optional)
- [ ] Gateway federation protocols
- [ ] Cross-platform content syndication
- [ ] Advanced privacy features (mix networks, dummy traffic)

Version 3.0 (Future):

- [ ] I2P integration (alternative to Tor)
- [ ] Pluggable transport support
- [ ] Advanced cryptographic features (threshold signatures, MPC)
- [ ] Zero-knowledge proofs for enhanced privacy
- [ ] Decentralized governance mechanisms

Release Timeline Summary

Version	Timeline	Status	Key Features
0.1 Alpha	Week 3	In Progress	OIP 0.9 DID system, create-but-don't-send
0.5 Beta	Week 8	Planned	WordPress + Tor + Name Resolution
0.8 RC	Week 13	Planned	Raspberry Pi image + Archive.org integration
1.0 Production	Week 18	Planned	Security audit + Public launch
1.5	+6 months	Future	Mobile/Desktop apps + Kiosk mode
2.0	+12 months	Future	Hardware keys + Advanced features

Total Development Time to 1.0: 18 weeks (~4.5 months) from Phase 1 kickoff

---

Conclusion

DWeb Server represents the culmination of nearly a decade of decentralized web development, bringing together the five key technologies from the 2016 DWeb Summit—DLOA/OIP, IPFS, WebTorrent, GUN, and DIDs—into a unified platform for truly anonymous, censorship-resistant publishing in an increasingly hostile information environment.

Why This Matters

- **For Whistleblowers**: Expose wrongdoing without fear of identification
- **For Journalists**: Publish stories that cannot be suppressed
- **For Researchers**: Share findings that challenge powerful interests
- **For Activists**: Organize and publish without surveillance

Why These Technologies Together

1. **Proven Foundation**: 9 years of evolution from DWeb Summit 2016 to production
2. **Standards-Based**: DID spec, W3C compliance, blockchain agnostic, open protocols
3. **Real-World Validation**: Caltech, Wyoming, Imogen Heap, WeAreChange, TimCast
4. **Historical Credibility**: Tim Berners-Lee called the original DLOA "thrilling" in 2016
5. **Zero Central Points of Failure**: All four DWeb pillars working together for the first time

What Makes This Different

- **Flexible Registration**: No registration if self-hosting; username + password if using gateway
- **One-Click Deploy**: Raspberry Pi images, not complex server setups
- **No Monthly Fees**: Self-hosted, peer-to-peer, no cloud dependencies
- **Gateway-Scoped Moderation**: Operators have control without global censorship
- **Local AI**: Alfred provides on-device Q&A with no cloud calls
- **Location Agnostic**: Metadata can live anywhere (not locked to single blockchain)

The Path Forward

With support from the Internet Archive, Freedom of the Press Foundation, and the broader DWeb community, we can build a platform that:

- Enables truly anonymous publishing
- Resists censorship without creating a lawless wasteland
- Remains accessible to non-technical users
- Costs nothing to operate from home
- Preserves information beyond any single entity's control

DWeb Server: *Publish anonymously. Index persistently. Distribute widely. Discover reliably. Resolve names without central control.*

Document Status: Draft for Review
Next Steps:

1. Stakeholder review (Internet Archive, FPF, DWeb community)
2. Begin Phase 1: OIP 0.9 DID system implementation (3 weeks)
3. Phase 2: WordPress plugin + Tor integration (5 weeks)
4. Phase 3: Name resolution (ENS + GNS) (5 weeks)
5. Phase 4: Raspberry Pi image and deployment tools (5 weeks)

Contact: amy@alexandria.io
Current Repository: https://github.com/DevonJames/oip-arweave-indexer