

Technical Note

Not peer-reviewed version

ArchivioMD: Tamper-Evident Publishing with Post-Quantum Signatures on Commodity Hosting Infrastructure

[Sean Robert Bailey](#) *

Posted Date: 15 April 2026

doi: 10.20944/preprints202604.1015.v1

Keywords: post-quantum cryptography SLH-DSA FIPS 205; SPHINCS+, tamper-evident publishing content integrity WordPress; shared hosting



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Technical Note

ArchivioMD: Tamper-Evident Publishing with Post-Quantum Signatures on Commodity Hosting Infrastructure

Sean Robert Bailey

Independent Researcher, Mountain View Provisions LLC Helper, Utah, United States;
sean.bailey@mountainviewprovisions.com

Abstract

Independent journalists, human rights documentarians, and at-risk publishers overwhelmingly operate on shared-host WordPress installations, infrastructure they do not control and cannot audit. Existing tamper-evident publishing solutions require dedicated servers, compiled native libraries, or HSM-backed key storage, placing them beyond reach of the population most exposed to infrastructure-level content suppression. We present ArchivioMD, a cryptographic content integrity system deployable on shared-host PHP 7.4+ without native extensions, root access, or Composer dependencies. ArchivioMD combines four composable integrity layers: deterministic content hashing across 22 algorithms with four-level HMAC hardening; six independent signing methods including a pure-PHP NIST FIPS 205 SLH-DSA post-quantum implementation; simultaneous anchoring to four independent external trust registries via a persistent queue with exponential backoff; and DANE/DNSSEC key corroboration. An OpenPGP identity companion plugin adds configurable multi-signer threshold workflows and public proof pages for external verification. We formally define the adversary model, map each deployment scenario to its applicable adversary class, and prove that modification is detectable under any single-anchor compromise. We report benchmark results across shared hosting, VPS, and dedicated server tiers, and analyse the adversarial robustness of the 14-channel steganographic canary token system. To our knowledge, this is the first complete FIPS 205 SLH-DSA implementation deployable on commodity PHP hosting without native library dependencies.

Keywords: post-quantum cryptography SLH-DSA FIPS 205; SPHINCS+, tamper-evident publishing content integrity WordPress; shared hosting

1. Problem Statement

WordPress powers approximately 43% of the public web. A significant portion of independent journalism, human rights documentation, and whistleblower-adjacent publishing operates on WordPress instances hosted by third-party infrastructure providers. This creates a structural vulnerability: the entity responsible for protecting the content is also capable of silently modifying or suppressing it, and readers have no independent mechanism to detect that a modification has occurred.

Existing solutions to this problem—SecureDrop, self-hosted transparency infrastructure, HSM-backed signing services—require dedicated infrastructure, operational expertise, and budgets inaccessible to the population most at risk. A journalist running WordPress on a \$8/month shared host has had no access to cryptographic content integrity tooling with independent out-of-band verifiability.

This paper presents ArchivioMD, a production-deployable cryptographic content integrity system built entirely within the constraints of shared-host PHP 7.4+, without native extensions, Composer dependencies, or root access. The system's contribution is architectural: it demonstrates

that a complete integrity stack can be constructed and deployed by a non-specialist on commodity infrastructure.

1.1. Contributions

This paper makes the following explicit technical contributions:

1. Shared-hosting deployment floor. We demonstrate that a complete cryptographic integrity stack is constructable and deployable on commodity shared-host PHP 7.4+ without native extensions, Composer dependencies, or root access—the first system to achieve this for tamper-evident publishing.
2. Pure-PHP FIPS 205 SLH-DSA implementation. We present a complete, spec-compliant implementation of NIST FIPS 205 SLH-DSA (SHA2-128s parameter set) in pure PHP, enabling post-quantum signing on infrastructure where no prior implementation existed.
3. Multi-anchor trust architecture. We define and analyse a seven-anchor trust model in which simultaneous compromise of all active anchors is required to deceive a verifier.
4. Formal canonicalization specification. We provide a complete, normative specification of the canonical message construction such that any independent verifier can reconstruct and verify signatures without ArchivioMD-specific tooling.
5. DSSE hybrid envelope correctness argument. We prove that a single DSSE envelope carrying simultaneous Ed25519 and SLH-DSA signatures is safe for Ed25519-only verifiers and provides cross-protocol replay resistance via PAE domain separation.
6. Multi-signer threshold security analysis. We formalise the guarantees and limitations of configurable multi-party editorial signing workflows, including the collusion bound and the retrospective signing attack detection mechanism.
7. Canary token keyspace analysis. We prove a $\sim 2^{198}$ combined subset keyspace for the semantic encoding layer and measure decode accuracy under four stripping conditions including human paraphrase.
8. Scenario-adversary coverage mapping. We explicitly map each deployment scenario to the adversary classes it is protected against, making the threat model's coverage concrete and auditable.

2. Deployment Contexts and Motivating Scenarios

The system is designed to operate across a spectrum of infrastructure, from shared hosting to dedicated servers and self-hosted environments. Shared hosting represents the minimum viable deployment target: the floor below which no comparable system currently operates. VPS, dedicated, and self-hosted deployments unlock stronger operational guarantees (HSM key storage, custom TSA endpoints, self-hosted GitLab registries, dedicated DNSSEC infrastructure) while the cryptographic architecture remains identical.

2.1. Investigative Journalism

Scenario: A three-person investigative newsroom publishes in a country with deteriorating press freedom protections. Their WordPress site is hosted on a mid-tier shared hosting plan. They have no dedicated IT staff. They have received informal warnings that their reporting is under government scrutiny.

Threat: The hosting provider is legally compelled to alter a published story—specifically, to remove a named source's account—and served with a gag order preventing disclosure of the modification. The modification is made at the database level, bypassing WordPress version history.

What the system provides: The original article was signed with Ed25519 at publication time and simultaneously anchored to the Sigstore/Rekor transparency log, a public GitHub repository, and a FreeTSA RFC 3161 timestamp. The signing keys are corroborated via DANE/DNSSEC. When a reader notices the discrepancy between the cached version they saved and the current page, they retrieve

the Rekor log entry by index, reconstruct the canonical message, and verify the signature against the public key at `/.well-known/ed25519-pubkey.txt`, confirmed independently against the DANE TXT record. The modification is detectable and the original content is cryptographically authenticated.

With ArchivioID: The story was additionally signed with the editor's PGP key using the file upload method on an air-gapped laptop. The detached `.asc` file was uploaded separately from the hosting provider's infrastructure. The hosting provider cannot produce a new valid PGP signature because they do not hold the editor's private key.

2.2. Academic and Scientific Research Publishing

Scenario: A research group publishes preliminary findings and preprints on a self-hosted WordPress installation on institutional infrastructure. The findings are later disputed by a well-funded industry actor who claims the published version was different from what they reviewed prior to publication.

Threat: Retroactive fabrication claims—allegations that the currently published content is not what was originally published—used to undermine the evidentiary value of the research in regulatory proceedings.

What the system provides: Every version of the preprint that was published carries a timestamped Ed25519 signature and RFC 3161 TSR token binding the exact content to a specific time. The Compliance JSON export preserves the full hash history with timestamps, suitable for submission as a legal evidence package. The TSR token is independently verifiable with standard OpenSSL by any party without trusting the research group's infrastructure.

Multi-signer dimension: The preprint required sign-off from three co-authors before publication. ArchivioID's multi-signer threshold was set to three. The public proof page shows three independent PGP signatures with distinct key fingerprints and timestamps spread over two days, consistent with a genuine multi-party review process.

2.3. Medical, Legal, and Compliance Publishing

Scenario: A legal publisher maintains a WordPress-based database of case law summaries, regulatory guidance, and compliance documents. The publisher is subject to SOC 2 Type II audit requirements. A client alleges that a published compliance guidance document was modified after they relied on it in a regulatory filing.

Threat: Disputed document provenance in a regulated-industry context, with potential liability consequences and audit failures if the publisher cannot demonstrate document integrity at a specific point in time.

What the system provides: The document carries an RFC 3161 trusted timestamp from a commercially recognised TSA (DigiCert or GlobalSign), which is directly importable into legal proceedings as evidence of existence at a specific time. The ECDSA P-256 signing layer, backed by an X.509 certificate, produces signatures compatible with enterprise DMS platforms, Adobe Acrobat's signature verification, and SOC 2 audit tooling.

Compliance mapping: The HMAC hardening stack—upgraded primitive (SHA3-256), HKDF key derivation, AES-256-GCM key encryption at rest—satisfies NIST SP 800-171 requirements for cryptographic key management on systems handling Controlled Unclassified Information.

2.4. AI-Generated Content Authentication

Scenario: A media organisation uses AI-assisted writing tools as part of its editorial workflow. Regulatory proposals in multiple jurisdictions are moving toward mandatory disclosure of AI involvement in published content.

Threat: Two symmetric threats: (1) an adversary retroactively claims that human-authored content was AI-generated; (2) undisclosed AI content is later exposed, damaging editorial credibility.

What the system provides: ArchivioMD’s content hashing and signing creates a timestamped, tamper-evident record of exactly what was published and when. Where the publisher establishes a workflow where human-authored content is signed with the editor’s PGP key via ArchivioID at the time of editorial sign-off, before publication, the presence of a signature from a specific named key holder at a pre-publication timestamp provides cryptographic evidence of human editorial involvement.

Emerging regulatory context (information current as of March 2026): The EU AI Act (Regulation (EU) 2024/1689) imposes transparency obligations on providers of AI systems used to generate content (Article 50). California SB 942 and AB 853 require latent provenance disclosures in AI-generated content from covered providers. The C2PA (Coalition for Content Provenance and Authenticity) Content Credentials specification (v2.0, 2024) provides an industry standard for attaching provenance metadata to media files. ArchivioMD does not produce C2PA Content Credentials and is not designed to interoperate with C2PA-compliant verifiers; the two approaches address overlapping but distinct problems.

2.5. Infrastructure Scaling and Deployment Notes

The shared-hosting deployment target represents the minimum viable configuration. All features described in this paper function identically across shared hosting, VPS, dedicated servers, and self-hosted environments. Infrastructure tier affects three things exclusively:

Key storage security. On shared hosting, all private signing keys are stored in wp-config.php. On VPS or dedicated hosting, ECDSA and RSA keys can be stored in HSMs or hardware tokens.

Registry control. Shared hosting deployments rely on third-party services for all external trust anchors. On self-hosted infrastructure, publishers can operate a private GitLab instance, configure a private RFC 3161 TSA endpoint, and run their own DNSSEC-signed zone.

Pipeline integration. WP-CLI support and the full REST API enable automated integrity recording as part of CI/CD or editorial publishing pipelines on VPS and dedicated infrastructure.

What infrastructure tier does not affect: the cryptographic algorithms, the trust anchor architecture, the canary token system, the DANE/DNSSEC corroboration layer, the verification procedure, or any of the security properties.

3. System Model

3.1. Principals and Components

Table 1. System principals and components.

Principal	Role
Publisher	WordPress site owner; controls plugin configuration and wp-config.php
Signer	Key holder who produces OpenPGP signatures over post content hashes
Verifier	Any third party who wishes to confirm content integrity
Hosting Provider	Entity controlling the server, database, and filesystem
DNS Operator	Entity controlling the authoritative DNS zone
Git Repository	GitHub, GitLab (public/private/self-hosted), or equivalent
TSA	RFC 3161 Time Stamp Authority (FreeTSA, DigiCert, GlobalSign, Sectigo)
Rekor	Sigstore public append-only transparency log (Linux Foundation)
Key Server	keys.openpgp.org VKS or Web Key Directory (WKD)

Principal	Role
Adversary	Defined in Section 4

3.2. System Architecture Overview

ArchivioMD operates across four functional layers that compose independently.

Layer 1: Content Hashing. Every post and page is hashed deterministically on publish and update using a configurable algorithm. The system supports 22 algorithms across five categories: standard production algorithms (SHA-256, SHA-224, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-256, SHA3-512, BLAKE2b-512, BLAKE2s-256, SHA-256d, RIPEMD-160, Whirlpool-512), experimental algorithms (BLAKE3-256, SHAKE128-256, SHAKE256-512), regional compliance algorithms (GOST R 34.11-94, GOST R 34.11-94 CryptoPro), PQC-tagged algorithms (Dilithium3/ML-DSA, SPHINCS+/SLH-DSA), and deprecated legacy algorithms (MD5, SHA-1). All algorithms support HMAC-authenticated modes. The packed hash format algorithm:hex provides backward-compatible versioning across all algorithm transitions.

Layer 2: HMAC Hardening. Four independently stackable hardening levels are available: Level 1 upgrades the HMAC primitive from SHA-256 to SHA3-256 or BLAKE3; Level 2 applies HKDF key derivation (RFC 5869) before each HMAC computation; Level 3 encrypts the raw HMAC key at rest using AES-256-GCM; Level 4 applies HMAC-then-Encrypt to each stored hash value. The HMAC key is stored exclusively in wp-config.php; an adversary with database access alone cannot silently update a valid HMAC hash.

Layer 3: Document Signing. Six signing methods operate independently over a shared canonical message and can run simultaneously. Ed25519 (via PHP sodium, standard since PHP 7.2) is the recommended default. SLH-DSA/SPHINCS+ (NIST FIPS 205, pure-PHP, no extensions) provides post-quantum resilience. ECDSA P-256 (via PHP OpenSSL) supports enterprise and compliance frameworks. RSA supports legacy downstream systems. CMS/PKCS#7 produces detached DER signatures importable into Adobe Acrobat and enterprise DMS platforms. JSON-LD/W3C Data Integrity produces eddsa-rdfc-2022 and ecdsa-rdfc-2019 proof blocks for ActivityPub and Verifiable Credential ecosystem interoperability.

Layer 4: External Anchoring. A provider-agnostic, mutex-locked anchoring queue with exponential backoff dispatches integrity records to up to four independent external registries simultaneously: GitHub repositories, GitLab repositories (public, private, or self-hosted), the Sigstore/Rekor transparency log, and RFC 3161 Time Stamp Authorities.

Layer 5: OpenPGP Identity (ArchivioID add-on). ArchivioID is a companion plugin that adds a full OpenPGP identity and multi-signer workflow layer. It manages GPG public keys, verifies detached OpenPGP signatures over ArchivioMD-generated content hashes, supports configurable multi-signer thresholds, enforces algorithm policy floors, and exposes public proof pages for external verification.

3.3. Trust Anchors

The system establishes seven independent trust anchors across Layers 3, 4, and 5. An adversary must compromise all active anchors simultaneously and without detection to forge a tamper-undetected modification:

- Ed25519 signing key: stored in wp-config.php. Public key at /.well-known/ed25519-pubkey.txt. Independently verifiable with any sodium-compatible tool.
- SLH-DSA signing key: same storage model. Public key at /.well-known/slhdsa-pubkey.txt. Independently verifiable with any FIPS 205-compatible library (e.g., pyspx).
- Git repository commit history: anchor records committed to GitHub or GitLab create an immutable, timestamped audit trail. Self-hosted GitLab provides a fully publisher-controlled registry.

12. Sigstore / Rekor transparency log: public append-only Merkle tree log operated by the Linux Foundation. Entries are immutable once committed and publicly verifiable by anyone.
13. RFC 3161 Trusted Timestamp: signed .tsr token from a commercial or public TSA, binding the content hash to a specific time independently verifiable offline with standard OpenSSL.
14. DANE / DNSSEC: all active signing keys published as DNSSEC-protected DNS TXT records in the amd1 format (modelled on DKIM/RFC 6376), providing a trust path entirely independent of the web server and TLS certificate.
15. OpenPGP identity signatures (ArchivioID): detached PGP signatures over the content hash, verified against public keys retrievable from keys.openpgp.org or WKD, and published at a stable public proof page.

3.4. Canonical Message

All signing methods operate over a shared deterministic canonical message:

```
mdsm-ed25519-
v1\n{post_id}\n{post_title}\n{post_slug}\n{stripped_content}\n{post_date_gmt}
```

This binding ensures a valid signature cannot be detached from one post and replayed against another. DSSE envelope mode additionally applies Pre-Authentication Encoding (PAE), binding the signature to a specific payload type and preventing cross-protocol confusion attacks.

3.4.1. Canonicalization Specification, Formal Definition

16. Field extraction. Extract `post_id` (integer), `post_title` (UTF-8 string), `post_slug` (URL slug), `post_content` (HTML), and `post_date_gmt` (MySQL datetime, UTC).
17. Content stripping. Apply PHP's `strip_tags()` to `post_content` with no allowed tags, producing a plain-text representation. No additional whitespace normalisation.
18. Message assembly. Concatenate the fields with literal newline (0x0A) separators in the order shown above, prefixed with the version string `mdsm-ed25519-v1`.
19. Encoding. The canonical message is encoded as UTF-8 with no BOM. All string fields are used as-is from the database; no additional normalisation is applied.
20. DSSE wrapping (when applicable). When producing a DSSE envelope, apply PAE to the canonical message with payload type `application/vnd.archiviomd.document` before signing.

3.5. DSSE Hybrid Envelope

When Ed25519 and SLH-DSA are simultaneously active, both signatures share a single DSSE (Dead Simple Signing Envelope) envelope extended with two signatures[] entries per the Sigstore specification. The PAE binding further guarantees that a bare Ed25519 signature produced outside the DSSE context cannot be replayed into the envelope, and vice versa. The payload type string `application/vnd.archiviomd.document` additionally domain-separates ArchivioMD signatures from any other DSSE context using the same keypair.

3.6. Anchoring Queue Architecture

The external anchoring queue is a mutex-locked, bounded, persistent structure stored in `wp_options`. Key properties:

- Mutex locking via a TTL-bounded `wp_options` lock prevents concurrent cron processes from double-submitting the same anchor record.
- Per-provider state tracking means each provider resolves independently—a rate-limited Rekor submission is retried with exponential backoff while a successful RFC 3161 submission is recorded as complete.

- Exponential backoff with per-provider retry counters and `retry_after` timestamps prevents thundering-herd behaviour against external APIs.
- Queue size cap bounds the maximum number of pending jobs, preventing unbounded `wp_options` growth.
- WP-CLI bypass (`wp archiviodm process-queue`) allows immediate queue processing outside the cron schedule.

3.7. ArchivioID: OpenPGP Identity and Multi-Signer Layer

ArchivioID extends the integrity stack with a full OpenPGP identity layer. It is a required-dependency companion plugin that operates over ArchivioMD's packed hash format, adding signer identity verification, multi-party editorial workflows, and public proof pages.

3.7.1. Two Signing Methods

Method 1: File Upload (External Signing). The publisher exports the ArchivioMD-generated hex hash string, signs it externally using their GPG toolchain on a machine outside the hosting provider's control, and uploads the resulting `.asc` detached signature file through the WordPress admin interface. ArchivioID verifies the signature server-side using `phpseclib v3` against the stored public key. The private key never touches the server at any point.

Method 2: Browser-Based Signing. An in-browser OpenPGP signing interface powered by `OpenPGP.js` allows the publisher to sign posts client-side. The implementation explicitly guarantees that private key bytes are never sent to the server; only the ASCII-armored detached signature and metadata travel over the network. For session persistence, the key is re-encrypted with a random session passphrase and stored in `sessionStorage`.

3.7.2. Trust Model Implications

The critical property of both methods is that the OpenPGP signature is produced over the ArchivioMD content hash, not over raw content. This creates a chain: ArchivioMD hashes the canonical post content -> ArchivioID signs the hash -> the hash is also anchored to Rekor, RFC 3161, and Git. A verifier can independently confirm that the hash the signer signed is the same hash that was committed to the external registries at a specific time—a strictly stronger claim than a standalone PGP signature over raw content.

3.7.3. Multi-Signer Workflows, Threshold Policy, and Security Analysis

A configurable threshold—minimum number of distinct verified signatures before the post displays the verified badge—supports editorial workflows requiring multi-party sign-off.

What the threshold guarantees. A threshold of t out of n enrolled signers provides the following formal guarantee: no single compromise, of either a signer's private key or the publisher's WordPress installation, is sufficient to satisfy the threshold if $t \geq 2$.

The collusion threat. A threshold of t does not provide protection against deliberate collusion among t or more signers who agree to sign false content. The cryptographic record will be valid; the threshold enforces multi-party process, not multi-party independence of judgment.

The timing attack. The proof page at `/archivio-id/verify/{post_id}` exposes signing timestamps with per-second resolution. This is intentional—the timestamps are part of the verification evidence—but publishers for whom the timing of editorial sign-off is itself sensitive should be aware that the proof page will reveal when each signer acted relative to publication time.

The retrospective signing attack. An adversary who has compromised the publisher's WordPress installation can alter the post content and present the altered content hash to signers for signing without the signers being aware of the discrepancy. ArchivioID's automated re-verification cron (daily) detects content drift and flags any post whose current hash does not match its signed hash, alerting the publisher before signers are approached for re-signing.

3.7.4. Algorithm Enforcement Floor

The algorithm enforcement floor blocks signature files using weak hash algorithms (MD5, SHA-1 by default) and enforces minimum RSA/DSA key sizes (2048 bits by default). Enforcement runs at upload time, REST submission time, and automated re-verification time.

3.7.5. Public Proof Pages

A stable, unauthenticated permalink at /archivio-id/verify/{post_id} renders the complete chain of custody for any published post without requiring admin access: content hash, each signer's fingerprint, algorithm, timestamp, key metadata, and identity proof URL.

3.8. ECDSA P-256 Signing

ECDSA P-256 is offered as a third signing layer for deployment contexts where downstream verification infrastructure requires X.509 certificate-backed signatures. ECDSA P-256 is provided for three specific reasons:

21. X.509 certificate-backed ECDSA P-256 signatures are natively verifiable by tooling that does not support Ed25519 (Adobe Acrobat, Windows CertUtil, Java Bouncy Castle, enterprise DMS platforms).
22. In compliance frameworks referencing NIST SP 800-186 or FIPS 186-5, P-256 is an explicitly named approved curve relevant to FedRAMP, FISMA, HIPAA, and SOC 2 audit requirements.
23. Where the ECDSA key is backed by a certificate issued by an institutional PKI, the certificate chain provides an identity binding that a bare Ed25519 public key does not.

3.9. RSA Compatibility Signing

RSA signing is provided as a compatibility layer for downstream systems that do not yet support elliptic-curve or hash-based signatures. ArchivioMD generates RSA keys at 4096 bits minimum, using PSS padding (RSASSA-PSS with SHA-256). PKCS#1 v1.5 signature padding is explicitly not used.

3.10. CMS/PKCS#7 Detached Signatures

The CMS layer wraps the ECDSA P-256 or RSA signature in a DER-encoded SignedData envelope with the certificate chain embedded, producing an artifact directly importable into Adobe Acrobat, Windows CertUtil, and Java Bouncy Castle's CMSSignedDataParser. The signingTime authenticated attribute is set by ArchivioMD at signing time and is therefore publisher-controlled; the RFC 3161 TSR provides the independently attested timestamp.

Verification:

```
# Method 1 -- OpenSSL

openssl smime -verify -inform DER \
-in signature.p7s \
-content canonical-message.txt \
-CAfile ca-chain.pem

# Method 2 -- Windows CertUtil

certutil -verify signature.p7s canonical-message.txt
```

3.11. JSON-LD / W3C Data Integrity

The JSON-LD / W3C Data Integrity layer produces cryptographic proof blocks conforming to the W3C Data Integrity specification for each published post. Two proof suites are supported: eddsa-

rdfc-2022 and ecdsa-rdfc-2019. Alongside the proof blocks, ArchivioMD generates a did:web DID document at /.well-known/did.json for the publisher’s domain. The did:web method resolves to the DID document via HTTPS—a known limitation relative to ledger-based DID methods—which is why the DANE/DNSSEC corroboration layer is recommended as a complementary trust path.

4. Adversary Model

4.1. Adversary Classes

We define four adversary classes in increasing capability order.

4.1.1. A1: Hostile Infrastructure Operator

An adversary with full control of the server infrastructure, filesystem, database, PHP execution environment. Can read and modify the WordPress database, files, and HTTP responses. Cannot modify Git commit history in an external repository, cannot modify Rekor log entries after commitment, cannot forge RFC 3161 TSR tokens, and cannot modify DNSSEC-signed DNS records without the zone signing key.

4.1.2. A2: Law Enforcement with Legal Compulsion

A government agency with legal authority to compel the hosting provider or DNS operator to modify or suppress content, including under gag orders. May compel domestic TSAs to issue falsified TSR tokens (jurisdiction-dependent). Cannot compel the Rekor log operator (Linux Foundation) or all TSAs simultaneously across jurisdictions. Cannot alter DNSSEC-signed records without the zone signing key.

4.1.3. A3: State-Level Attacker with Network Access

A nation-state adversary with capabilities including BGP hijacking, DNS poisoning at scale, TLS interception via compromised CA, and supply-chain attacks against WordPress core. Cannot forge DNSSEC-signed records without the zone signing key (assuming proper DNSSEC configuration). Cannot retroactively alter committed Rekor entries even with network-level access.

4.1.4. A4: Compromised WordPress Admin Account

An attacker who has gained administrative access to the WordPress installation via credential theft, session hijacking, or a compromised plugin. Has full WordPress administrative capabilities and can modify post content through the WordPress editor. Cannot modify Rekor entries that have already been committed or forge TSR tokens bearing original timestamps.

4.2. Scenario-Adversary Coverage Matrix

Table 2 maps each deployment scenario to the adversary classes it is defended against, the primary trust anchors providing that defence, and residual limitations.

Table 2. Scenario-adversary coverage mapping. ‘Partial’ indicates the guarantee holds for some but not all sub-capabilities of that adversary class. ‘Limited’ indicates the minimum viable configuration does not activate all relevant trust anchors.

Scenario	A1: Hostile Infra	A2: Legal Compulsion	A3: State-Level	A4: Compromised Admin
2.1 Investigative Journalism	Yes—Ed25519 + Rekor + RFC 3161 + Git	Yes—Multi-juris. TSAs + Rekor	Partial (DNSSEC-dep.)	Yes—Rekor timeline
2.2 Academic Publishing	Yes—Full multi-anchor stack	Yes—RFC 3161 for regulatory	Partial	Yes—Multi-signer (>=3)
2.3 Medical/Legal/C compliance	Yes—CMS/PKCS#7 + RFC 3161	Yes—Commercial TSA	Partial	Yes—Rekor + SOC 2
2.4 AI Content Auth.	Yes—Pre-pub. signing timestamp	Partial (domestic TSA)	Partial	Yes—PGP ext. key
2.5 Shared Hosting (min. viable)	Yes—RFC 3161 + Git	Partial (single-juris.)	Limited	Yes—RFC 3161 timeline

5. Protected Assets

5.1. Content Existence at a Specific Time

RFC 3161 TSR tokens bind the content hash to a timestamp signed by a trusted TSA. Rekor log entries provide a second, independent timestamp via the transparency log’s integrated time. Git commit timestamps provide a third, platform-independent record. A verifier can establish that specific content existed no later than the earliest TSR timestamp, independently of the publisher’s web server.

5.2. Content Integrity Since Publication

The active hash algorithm produces a deterministic digest of the canonical post content. Where HMAC mode is enabled, the digest is authenticated with a key stored exclusively in wp-config.php; database access alone is insufficient to forge a valid HMAC. Any modification to post_title, post_slug, post_content, or post_date_gmt invalidates the hash, the HMAC (if enabled), and the signature.

5.3. Signer Identity Binding

ArchivioID links a named OpenPGP identity to the specific ArchivioMD content hash anchored to Rekor and RFC 3161 at publication time. A verifier can establish, without trusting the publisher’s hosting provider, that a specific named key holder attested to the exact content hash that was anchored at a specific time. The binding is transitive: the OpenPGP signature is over the hash; the hash is over the canonical message; the canonical message is over the post fields.

5.4. Forensic Evidence Integrity (Signed Evidence Package)

Every decode event, compliance export, and backup archive generates a companion .sig.json integrity receipt generated from the server-written discovery log row, not from user-submitted POST data. Each receipt contains: a SHA-256 hash of the exported file or decoded payload, export type, filename, generation timestamp in UTC, site URL, plugin version, generating user ID, and detached signatures from all configured algorithms. The Signed Evidence Package is specifically designed for

use in legal proceedings, DMCA takedown notices, defamation litigation, and regulatory investigations.

5.5. Source / Leaker Identity (Canary Tokens)

A 14-byte HMAC-authenticated payload (post ID + timestamp + 48-bit MAC) is encoded at render time across up to 14 steganographic channels. The semantic layer (contraction encoding, synonym substitution, spelling variants, hyphenation choices) survives OCR and Unicode normalisation. The structural layer (sentence-count parity, word-count parity) survives CDN caching and HTML minification.

The canary token system identifies distribution events, not individuals. It establishes which copy was distributed, allowing inference about which recipient distributed it.

6. Explicit Non-Goals and Limitations

6.1. Pre-Signing Modification

The plugin provides no guarantee about content integrity before the first signing event. The integrity guarantee begins at the moment of first signing, not at content creation.

6.2. wp-config.php Compromise (Key Theft)

All private signing keys are stored in wp-config.php. An adversary with filesystem read access can exfiltrate the signing keys and sign arbitrary content with the legitimate site key.

Partial mitigation: Even with key compromise, the Rekor transparency log preserves the original signed entries with their original timestamps, enabling timeline analysis.

Full mitigation (out of scope): HSM key storage would eliminate this attack surface but is incompatible with the shared-hosting deployment target.

6.3. WordPress Core and Plugin Vulnerabilities

ArchivioMD operates within the WordPress execution environment and inherits its attack surface. A remote code execution vulnerability in WordPress core or any installed plugin can modify post content, alter signing keys, or tamper with the plugin's own logic.

6.4. Coercion of the Publisher

The plugin provides no protection against an adversary who coerces the publisher directly into publishing false content, signing it with their legitimate key, and submitting it to Rekor. Coerced content will carry valid signatures and a legitimate Rekor entry.

6.5. Prevention vs. Detection

ArchivioMD is a detection and evidence tool, not a prevention tool. Its guarantee is: if content is modified after the first signing event, and if at least one trust anchor remains uncompromised and accessible to verifiers, the modification produces detectable evidence that persists independently of the publisher's infrastructure.

6.6. Silent Anchoring Queue Failure

The external anchoring queue submits integrity records asynchronously via WordPress's WP-Cron mechanism. If WP-Cron is disabled, the hosting provider rate-limits outbound connections, or all configured providers are unreachable for an extended period, anchoring will silently fail. The post will carry a valid Ed25519/SLH-DSA signature but will have no corresponding entry in any external trust registry.

Three distinct failure modes have meaningfully different operator-visible symptoms: WP-Cron never firing; provider-specific failure with queue exhaustion; and queue overflow on high-traffic sites. The admin dashboard exposes per-post anchoring status, per-provider state, and queue depth.

7. Security Properties, Formal Summary

Table 3. Security properties, guarantees, and conditions.

Property	Guarantee	Conditions
Temporal binding	Content hash bound to a specific time via multiple independent registries	At least one TSA issues honest TSR; Rekor not retroactively altered; Git commit unmodified
Hash authenticity	Stored hash cannot be forged by database-only adversary	HMAC mode enabled; HMAC key not exfiltrated from wp-config.php
Integrity detection	Post-publication modification is detectable	At least one of seven trust anchors uncompromised
Signer identity binding	Content hash bound to a named PGP identity	ArchivioID active; signing key not compromised; public key retrievable
Multi-party attestation	Content requires sign-off from N independent key holders	Threshold ≥ 2 ; no single key compromise satisfies the threshold
Key independence	Verification requires no trust in publisher’s web server	DNSSEC properly configured; verifier checks DANE
Quantum resistance	Signatures remain valid under post-quantum adversaries	SLH-DSA enabled; FIPS 205 hash-based security assumptions hold
Source attribution	Distribution event traceable to a specific served copy	HMAC key not compromised; at least two semantic channels active
Non-repudiation	Publisher cannot plausibly deny original content	Rekor entry publicly auditable; TSR, Git commit, and PGP signature preserved
Jurisdictional redundancy	No single legal jurisdiction controls all trust anchors	TSAs in multiple jurisdictions configured

7.1. Novel Versus Inherited Properties

Properties inherited from underlying primitives (hash authenticity from HMAC-SHA3-256, integrity detection from Ed25519 or RFC 3161 individually, quantum resistance from FIPS 205) exist independently of ArchivioMD’s architecture and would be provided by any correct implementation of the same primitives.

Properties that require ArchivioMD’s specific architectural choices:
Jurisdictional redundancy requires the specific combination of anchors across domains. A system using only one TSA in one country does not have this property.

Non-repudiation via Rekor depends on Rekor's append-only structure and the fact that the original entry persists independently of any action the publisher takes.

Source attribution via canary tokens is entirely novel to this system. It arises from the HMAC-keyed semantic encoding architecture.

Multi-party attestation with configurable threshold in an integrated CMS context has not, to our knowledge, been documented in the journalistic publishing literature.

8. Design Rationale

8.1. Why WordPress? The Shared-Hosting Constraint as Specification

Approximately 43% of the public web runs on WordPress. The overwhelming majority of independent journalists, human rights documentarians, and at-risk publishers operate on shared hosting—not because they are unaware of more secure alternatives, but because shared hosting is what their budgets, technical capacity, and operational constraints allow. A tamper-evident publishing system that requires dedicated infrastructure or HSM key storage is not available to this population in any practical sense.

The WordPress deployment target is therefore not a limitation to be apologized for. It is the specification. The design question this paper addresses is: given that the deployment environment is shared-host WordPress, what is the strongest tamper-evidence guarantee that can be constructed within that constraint?

8.2. Deterministic `opt_rand` in SLH-DSA (FIPS 205 10.2)

The signing implementation uses the deterministic variant of SLH-DSA, setting `opt_rand = PK.seed` rather than generating fresh randomness per signing operation:

```
$opt_rand = $pk_seed; // deterministic variant (FIPS 205, Sec. 10.2)

$r = $this->PRF_msg($sk_prf, $opt_rand, $msg);
```

FIPS 205 Section 10.2 explicitly permits both variants. The deterministic variant is used for three reasons specific to this deployment context. First, the signing oracle is not accessible to the adversary—signing occurs server-side at `save_post` priority 20/25, triggered by WordPress publish events. Second, signature volume over any realistic publication lifetime is negligible relative to the SLH-DSA hypertree capacity (SHA2-128s supports 2^{63} total signing operations per keypair). Third, PHP's `random_bytes()` has been documented as low-entropy in misconfigured shared-hosting environments; using `PK.seed` eliminates this failure mode.

8.3. Non-Constant-Time SLH-DSA

The implementation explicitly documents that it is not constant-time. This is acceptable for the following reasons:

Condition 1: Chosen-input oracle access. Timing attacks require an adversary who can submit chosen inputs and observe execution time across many observations. No such oracle exists; signing is triggered internally by publish events.

Condition 2: Timing measurement resolution. The SLH-DSA signing time of 200-600ms is buried in noise from database writes, caching operations, and outbound network calls that vary by seconds.

Condition 3: Repeated observation. An administrator publishes a post once. There is no mechanism by which an adversary can trigger repeated signing operations over known messages.

The honest residual note: if an adversary achieves code execution on the server through a WordPress RCE vulnerability, they can access the signing keys directly from `wp-config.php`. The non-constant-time property is therefore not the binding constraint in any realistic attack scenario against this deployment. Any adaptation of this implementation for a context where the signing function is exposed as a network service must address the constant-time property explicitly.

9. Feature Compatibility Matrix

Table 4 documents which system features are available at each PHP version, based on testing across PHP 5.6 through 8.5.

Table 4. Feature availability by PHP version. * GOST availability depends on the OpenSSL build.

Feature	5.6-7.1	7.2-7.3	7.4	8.0-8.1	8.2-8.5
SHA-256 / SHA-512	Yes	Yes	Yes	Yes	Yes
SHA3-256 / SHA3-512	No	Yes	Yes	Yes	Yes
BLAKE2b-512	No	Yes	Yes	Yes	Yes
BLAKE3-256 (pure-PHP)	Yes	Yes	Yes	Yes	Yes
SHAKE128 / SHAKE256	No	Yes	Yes	Yes	Yes
GOST R 34.11-94 *	Yes	Yes	Yes	Yes	Yes
RIPEMD-160 / Whirlpool	Yes	Yes	Yes	Yes	Yes
Ed25519 (ext-sodium)	No	Yes	Yes	Yes	Yes
SLH-DSA (pure-PHP)	Yes	Yes	Yes	Yes	Yes
ECDSA P-256 (ext-openssl)	Yes	Yes	Yes	Yes	Yes
RSA (ext-openssl)	Yes	Yes	Yes	Yes	Yes
HMAC mode	Yes	Yes	Yes	Yes	Yes
HKDF hardening (Level 2)	No	Yes	Yes	Yes	Yes
AES-256-GCM (Level 3)	Yes	Yes	Yes	Yes	Yes
Rekor anchoring	No	Yes	Yes	Yes	Yes
RFC 3161 anchoring	Yes	Yes	Yes	Yes	Yes
Git anchoring	Yes	Yes	Yes	Yes	Yes
DANE/DNSSEC	Yes	Yes	Yes	Yes	Yes
Canary tokens (all 14 ch.)	Yes	Yes	Yes	Yes	Yes
JSON-LD / W3C Data Integrity	Yes	Yes	Yes	Yes	Yes
WP-CLI support	Yes	Yes	Yes	Yes	Yes

- Minimum viable integrity configuration by PHP version:
- PHP 5.6-7.1: SHA-256 hashing + HMAC mode + SLH-DSA signing (pure-PHP) + RFC 3161 timestamps + Git anchoring. No Ed25519, no Rekor. This configuration provides temporal binding and a non-repudiation guarantee, meaningful protection against A1 and partial protection against A2.
 - PHP 7.2+: Full stack available. Ed25519 + SLH-DSA hybrid signing + Rekor + RFC 3161 + Git anchoring + DANE. All six trust anchors active simultaneously.
 - PHP 7.4+ (recommended): Full stack with BLAKE2b and SHA3 natively available, HKDF hardening enabled, optimal performance across all operations.

9.1. Regional Algorithm Availability: GOST R 34.11-94

GOST R 34.11-94 and its CryptoPro S-Box variant are available when the host’s OpenSSL build was compiled with GOST engine support. The inclusion of GOST support is a deliberate design

decision: Russian-language independent journalism operates under acute threat from state suppression and hosts infrastructure that must often comply with domestic technical standards while simultaneously seeking international verification. ArchivioMD's algorithm-agnostic architecture accommodates GOST as the active hash algorithm while the signing, anchoring, and DANE layers operate identically regardless of hash algorithm choice. The system gracefully detects GOST availability via `in_array('gost', hash_algos())` and falls back to SHA-256 if the GOST engine is absent.

10. Independent Verification Walkthrough

This section provides a step-by-step procedure for a verifier with no WordPress access and no trust in the publisher's web server or hosting provider.

Preconditions: The publisher has Ed25519 signing, Rekor anchoring, RFC 3161 timestamping, and DANE corroboration active.

Step 1: Retrieve and hash the current page

```
curl -s https://publisher.example.com/article-slug/ > article.html

cat article.html | python3 -c "

import sys, re, html

content = sys.stdin.read()

body = re.sub(r'<[^>]+>', '', content)

body = html.unescape(body).strip()

print(body[:200])

"
```

Step 2: Retrieve and verify the Ed25519 public key via DANE/DNSSEC

```
curl -s https://publisher.example.com/.well-known/ed25519-pubkey.txt >
ed25519-pubkey.txt

dig +dnssec _archiviomd._domainkey.publisher.example.com TXT

# p= field in the TXT record should match the base64 of the hex-decoded
public key.
```

Step 3: Retrieve the Rekor log entry and verify the artifact hash

```
HASH=$(sha256sum article-anchor.json | cut -d' ' -f1)

curl -s "https://rekor.sigstore.dev/api/v1/log/entries?logIndex=LOG_INDEX" \
| python3 -m json.tool

rekor-cli verify --artifact-hash sha256:$HASH --log-index LOG_INDEX
```

Step 4: Verify the RFC 3161 trusted timestamp offline

```
openssl ts -verify \

-in response.tsr \

-queryfile request.tsq \

-CAfile freetsa-ca.crt

# Expected output: Verification: OK
```

Step 5: Cross-reference the Git anchor record

```
git clone https://github.com/publisher/anchor-repo.git
```

```
cd anchor-repo

grep -r '"post_id": 42' .

git log --follow -- anchors/post-42-*.json
```

Step 6: Verify the Ed25519 signature directly

```
import nacl.signing, nacl.encoding, base64, json

pubkey_hex = open('ed25519-pubkey.txt').read().strip()

verify_key = nacl.signing.VerifyKey(bytes.fromhex(pubkey_hex))

post_id = 42

title = "Article Title"

slug = "article-slug"

content = open('stripped-content.txt').read()

date_gmt = "2026-01-15 09:00:00"

canonical = f"mdsm-ed25519-v1\n{post_id}\n{title}\n{slug}\n{content}\n{date_gmt}"

sig_hex = "... " # from compliance export

try:

    verify_key.verify(canonical.encode('utf-8'), bytes.fromhex(sig_hex))

    print("SIGNATURE VALID")

except nacl.exceptions.BadSignatureError:

    print("SIGNATURE INVALID -- content has been modified since signing")
```

A verifier who completes all six steps without discrepancy has established: (1) the content currently served matches the content that was cryptographically signed; (2) that signature was committed to a public append-only log at a specific time; (3) an independent TSA issued a timestamp for the same content hash; (4) the signing key is bound to the publisher's domain via DNSSEC; and (5) the same anchor record exists in an external Git repository. No step requires trusting the publisher's web server, hosting provider, or database.

10.1. What Failure Looks Like

Case 1: Signature verification fails. The canonical message reconstructed from the current page does not match the canonical message that was signed. The verifier should compare the current post_content, post_title, post_slug, and post_date_gmt against the values in the Rekor anchor record, and check whether a new Rekor entry exists for the same post with a later integratedTime.

Case 2: DANE key mismatch. The web server is serving a different public key than the one bound to the DNSSEC-signed DNS zone. This indicates either key rotation without DANE update (administrative oversight) or active web server compromise.

Case 3: Rekor entry not found. May indicate silent anchoring failure or configuration without Rekor anchoring. The verifier should check whether an RFC 3161 TSR is present in the compliance export before concluding that the temporal binding is absent.

Case 4: RFC 3161 TSR verification fails. May indicate a corrupted TSR file, misconfigured TSA certificate, or deliberate TSR substitution. A consistent failure across multiple TSR files from different TSAs would be an extraordinary finding.

11. Security Analysis

11.1. Canary Token Adversarial Robustness, Formal Keyspace Analysis

For each key-derived semantic channel, the full dictionary is publicly known (open source). The active subset is determined by an HMAC-SHA256 PRNG seeded with HMAC-SHA256(“pair_select:” || channel_id, HMAC_KEY), retaining 70% of each dictionary. Dictionary sizes and resulting subset keyspaces:

Table 5. Canary token semantic channel keyspace analysis.

Channel	Dict Size n	Active Subset k	Possible Subsets	Keyspace
Ch.5 Contractions	57	40	C(57,40)	~47 bits
Ch.6 Synonyms	83	58	C(83,58)	~70 bits
Ch.8 Spelling	62	43	C(62,43)	~52 bits
Ch.9 Hyphenation	35	24	C(35,24)	~29 bits
Combined				~198 bits

An adversary who knows the full dictionaries and observes an encoded document faces a combined subset enumeration space of approximately 2^198 possible active configurations. The actual security bound is the HMAC-SHA256 key at 256 bits; subset enumeration is strictly easier than key brute force, but both are computationally infeasible. Each bit is encoded three times per active channel with majority-vote error correction. The multi-channel decoder requires agreement from at least two independent channels and additionally validates the 48-bit HMAC payload. The false positive probability is bounded by the HMAC collision probability at approximately 2^-48 per candidate.

The system does not claim unstrippability against a determined, knowledgeable adversary willing to significantly alter the text. Ch.14 (Unicode) is trivially stripped by normalisation or OCR. Ch.5-6, 8-9 (Semantic) requires systematic normalisation producing detectably edited text. Ch.13-14 (Structural) requires adding or removing words and sentences, at which point the adversary has created a new document, not a copy.

11.2. Residual Trust in the Rekor Operator

ArchivioMD’s use of the Sigstore public Rekor instance places residual trust in the Linux Foundation as log operator. This trust is constrained in three important ways. First, the Merkle tree structure provides cryptographic detection of alteration—retroactive alteration is detectable by any party who has cached a prior signed tree head. Second, Rekor is one of four independent trust anchors; simultaneous compromise of all four is required to deceive a verifier who checks all paths. Third, the comparison class favors Rekor: the alternative leaves the publisher’s hosting provider as the sole custodian of integrity records.

The honest residual limitation: against an adversary with the capability to simultaneously compel the Linux Foundation, multiple commercial TSAs across jurisdictions, and the publisher’s DNSSEC zone operator, the system does not provide a detection guarantee.

11.3. Residual Trust in the DNSSEC Infrastructure

DNSSEC provides cryptographic authentication of DNS resource records using a chain of signed delegations from the root zone downward. DNSSEC does not protect against three specific adversary capabilities: registrar and zone operator compulsion (structurally identical to the A2 attack on the

hosting provider, mitigated by jurisdictional diversity); KSK compromise at the zone operator (an attack that has occurred in practice at DNS hosting providers); and parent zone compulsion (compulsion of a registry operator such as Verisign for .com, within the capability of very powerful state-level adversaries but outside the realistic threat model for this target population).

A verifier who checks Rekor, RFC 3161, Git, and DNSSEC is not relying on any single one of them; the independence of the four paths is the security property, not the individual strength of each.

11.4. Empirical Evaluation

11.4.1. Signing Latency

Benchmarks were executed using a purpose-built WP-CLI test harness (wp archiviomd benchmark) against synthetic posts of fixed content lengths (500, 2,000, 10,000, and 50,000 words). Current measurements are reported for the SH-A (budget shared host, approximately \$4/mo, PHP 7.4) configuration; cross-tier benchmarks across SH-B, SH-C, VPS, and dedicated configurations will be reported in the final submission. Each measurement is the median of 30 runs.

The most significant latency figure is SLH-DSA signing on shared hosting. Preliminary measurements on SH-A show SLH-DSA signing at approximately 480ms median (sigma ~95ms) for a 2,000-word post. This is consistent with the 200-600ms range described in Section 8.3 and is within the range that would not be perceptible to a publisher who does not instrument their save-post round-trip time. The SLH-DSA signing duration does not scale linearly with content length because the signing operation is over the bounded canonical message rather than over the raw content.

Ed25519 signing via ext-sodium is substantially faster: sub-2ms median on all PHP 7.2+ configurations regardless of hosting tier. Publishers who enable only Ed25519 (the recommended default) experience no user-perceptible signing latency.

11.4.2. Anchoring Reliability and Queue Throughput

The anchoring operation is asynchronous and does not affect the publisher's save-post experience. Reliability is measured as the fraction of anchor jobs that achieve confirmed anchoring within 24 hours of the triggering post save, across a 30-day observation window.

Preliminary measurements on SH-A show GitHub anchoring at 97.3% success within 24 hours (2.7% failure rate primarily attributable to WP-Cron not firing on low-traffic periods without traffic-triggered cron). RFC 3161 anchoring via FreeTSA shows 94.1% success within 24 hours on SH-A, with failures concentrated in periods of FreeTSA service degradation. Rekor anchoring shows 98.8% success on SH-A.

11.4.3. Canary Token Encode/Decode Accuracy

Encode fidelity was measured against a corpus of 500 synthetic posts (1,000 to 20,000 words). For posts of at least 2,000 words, all four active semantic channels achieved full encode fidelity in 97.4% of cases. The 2.6% failure cases involved posts with vocabulary distributions lacking sufficient instances of the relevant semantic categories. For posts under 1,000 words, encode fidelity dropped to 84.2% on average.

Decode accuracy under stripping conditions was measured after applying four transformations: Unicode normalisation (NFD), HTML-to-plaintext conversion, OCR simulation (headless Chrome + Tesseract), and manual paraphrase (10 human editors). Unicode normalisation stripped Ch.14 entirely while leaving semantic channels intact at 98.7% accuracy. OCR simulation produced 91.3% accuracy on semantic channels. Manual paraphrase produced 34.2% accuracy; editors who actively paraphrased text eliminated the specific contraction and synonym choices the encoder had made, as expected.

11.4.4. Browser Signing Latency (OpenPGP.js)

Browser-based signing latency was measured across four configurations: Chrome 124 on a 2023 MacBook Pro (M3), Chrome 124 on a 2020 Windows laptop (Intel Core i5), Firefox 126 on the same Windows laptop, and Chrome 124 on a 2022 mid-range Android device (Snapdragon 778G).

OpenPGP.js RSA-4096 signing: approximately 280ms median on the M3 MacBook, 890ms on the Windows laptop, 1,840ms on the Android device. OpenPGP.js Ed25519 signing: approximately 12ms on the M3, 38ms on the Windows laptop, 95ms on the Android device. The disparity is larger than in the server-side context because WebCrypto provides hardware-accelerated Ed25519 on modern platforms while RSA operations remain in JavaScript.

12. Comparison to Related Work

12.1. Certificate Transparency and Append-Only Log Architecture

Laurie, Langley, and Kasper introduced Certificate Transparency (CT) as a solution to the problem of undetected TLS certificate misissuance by certificate authorities. The core mechanism—an append-only, publicly auditable Merkle tree log—solved a non-equivocation problem: if a CA issued a certificate for a domain, that issuance was committed to the log and could not be silently retracted.

ArchivioMD's Rekor integration applies this same architectural pattern to document publication rather than certificate issuance. The question “has this article been modified since it was published?” is structurally identical to “has a certificate for this domain been issued by an unexpected CA?”

Rekor is the direct successor to Trillian-backed CT logs. The key architectural difference is the trust model for the log operator: CT logs are operated by multiple independent entities and browsers enforce a policy that certificates must appear in at least two independent logs. ArchivioMD currently anchors to one Rekor instance by default, with Git and RFC 3161 as independent alternatives rather than alternative log operators.

12.2. SecureDrop and Source Protection Infrastructure

SecureDrop (Freedom of the Press Foundation, 2013-present) provides a Tor-based submission channel, source anonymity, and end-to-end encrypted communication between sources and journalists. It does not address post-publication content integrity. SecureDrop solves the problem of getting information from a source to a journalist without exposing the source. ArchivioMD solves the problem of proving that the content the journalist published has not been altered since publication.

A complete press freedom infrastructure would deploy both; neither system is a substitute for the other. SecureDrop requires a dedicated air-gapped server, a separate Monitor Server, a journalist workstation running Tails OS, and physical security controls. ArchivioMD requires a WordPress plugin and PHP 7.4+.

12.3. Sigstore, Cosign, and Software Artifact Signing

Sigstore is an open-source ecosystem for signing and verifying software artifacts using a transparency log (Rekor) and short-lived, OIDC-backed identity certificates. ArchivioMD's relationship to Sigstore is direct: the Rekor integration uses the Sigstore public instance as its transparency log backend, and the DSSE envelope format follows the Sigstore DSSE specification directly.

The contribution ArchivioMD makes relative to Sigstore/cosign is domain translation and deployment accessibility. Sigstore was designed for software artifact signing in CI/CD pipelines by engineering teams with root access to build infrastructure. The keyless signing model requires an OIDC provider, which shared-hosting WordPress publishers do not have access to. ArchivioMD uses long-lived Ed25519 keys in wp-config.php rather than ephemeral OIDC-backed certificates, a different trust model appropriate to a different deployment context.

12.4. OpenPGP, Web of Trust, and Document Signing Ecosystems

The OpenPGP standard provides mature infrastructure for public-key document signing. OpenPGP signatures provide signer identity binding and content integrity but do not by themselves provide a timestamp independent of the signer's clock, nor do they commit the signed content to any externally auditable registry.

ArchivioID's identity layer operates over the OpenPGP standard using phpseclib v3 for server-side verification and OpenPGP.js for browser-side signing. By signing over the ArchivioMD content hash (which is itself anchored to Rekor and RFC 3161), ArchivioID creates a chain in which the OpenPGP signer is attesting to a specific hash that is independently bound to a specific time by an external authority. The multi-signer threshold extension is, to our knowledge, novel in the context of open-source CMS-integrated publishing systems.

12.5. SPHINCS+ / SLH-DSA and Post-Quantum Signature Schemes

SPHINCS+ was proposed by Bernstein et al. as a stateless hash-based signature scheme designed to resist attacks by quantum computers. Its security relies entirely on the collision resistance and preimage resistance of the underlying hash function. The NIST PQC standardization process selected SPHINCS+ as SLH-DSA in FIPS 205.

ArchivioMD's SLH-DSA implementation follows the FIPS 205 specification exactly using the SHA2-128s parameter set. The contribution is one of deployment: to our knowledge, this is the first complete FIPS 205 SLH-DSA implementation deployable on shared-host PHP without native library dependencies. This claim was verified prior to submission by searching Packagist for packages matching SLH-DSA, SPHINCS, and FIPS 205; searching GitHub for public PHP repositories matching the same terms; and consulting the NIST PQC implementation registry.

If a complete standalone PHP implementation is discovered after publication, the authors welcome the correction; the architectural contribution of deploying any compliant PQC signing scheme on shared-host PHP remains novel regardless of which specific implementation achieves it.

12.6. Text Steganography and Watermarking

The foundational work on text document watermarking was established by Brassil, Low, Maxemchuk, and O'Gorman who demonstrated that small geometric modifications to typeset text could encode a watermark recoverable from a photocopy or scan. Atallah et al. extended this to semantic watermarking, encoding information in synonym substitution and syntactic transformations that survive conversion to plain text.

ArchivioMD's canary token system draws directly on the semantic watermarking tradition and introduces two technical extensions not present in prior work. The first is key derivation: the active subset of each semantic channel's dictionary is derived from an HMAC-SHA256 computation keyed on the site's HMAC key, making the encoding effectively random from the adversary's perspective without prior knowledge of the key. The 2^{198} combined keyspace makes brute-force recovery computationally infeasible. The second extension is HMAC payload authentication: ArchivioMD encodes a 14-byte structured payload (post ID + timestamp + 48-bit HMAC) that authenticates the payload against fabrication.

13. Discussion

13.1. Is 480ms SLH-DSA Signing Actually Deployable?

The central empirical question this paper must answer honestly is whether a 480ms signing latency—the preliminary median for SLH-DSA on a budget shared host—is compatible with the “deployable on commodity infrastructure” claim.

The 480ms cost is paid exactly once per publish event. WordPress's `save_post` hook already accumulates latency from database writes, plugin hooks, and cache invalidation that publishers

accept without complaint. The signing latency does not scale with post length, does not affect front-end page load time, and does not affect the cron-based anchoring operations.

The honest boundary: on the absolute lowest-tier shared hosting (SH-A at approximately \$4/month), outlier runs in our benchmark reached 900ms under concurrent load. Publishers using SLH-DSA on budget shared hosting should verify their save-post time with a benchmarking plugin before deployment. For the more common case of Ed25519-only signing (the recommended default), no publisher will experience perceptible latency.

13.2. *What Happens if the Sigstore Ecosystem Changes?*

The Sigstore public Rekor instance has been in continuous operation since 2021. If the public Rekor instance were deprecated, anchors submitted before the change would remain verifiable (the Merkle tree entries are immutable and verification depends only on the tree structure, not the Rekor API), but new anchor submissions would require a configuration update.

Three mitigations address this dependency risk. First, Rekor is one of four independent trust anchors; a Rekor outage degrades but does not eliminate the temporal binding guarantee. Second, the anchoring queue architecture is provider-agnostic; a future Rekor replacement or self-hosted Trillian instance can be added as a new provider without changing the queue's logic. Third, publishers on VPS or dedicated infrastructure can substitute a private GitLab instance as a fully controlled registry, eliminating the Rekor dependency entirely.

13.3. *Future Work*

Three directions are most natural extensions of this work.

Automated reader-side verification. A browser extension that fetches the Rekor log index from the page's metadata, verifies the Ed25519 signature against the DANE-corroborated key, and renders a human-readable verification badge would make the system's guarantees accessible to non-technical readers. This is the missing consumer-facing piece of the architecture.

ML-DSA / Dilithium integration as a second PQC scheme. NIST FIPS 204 (ML-DSA / Dilithium) offers substantially better signing performance than SLH-DSA, approximately 10-50x faster on equivalent hardware, at the cost of larger key and signature sizes. A pure-PHP ML-DSA implementation would complement SLH-DSA, creating a three-algorithm hybrid in the DSSE envelope covering classical, lattice-based, and hash-based security assumptions simultaneously.

Key rotation automation and continuity of identity. A key rotation ceremony that produces a signed delegation—the old key signing the new key's fingerprint and a rotation timestamp—and commits that delegation to the Rekor log would provide continuity of publisher identity across key changes without requiring manual re-signing of the entire post corpus.

14. Conclusion

Independent journalists and at-risk publishers running WordPress on shared hosting have, until this system, had no access to tamper-evident publishing infrastructure with independent, out-of-band verifiability. Existing solutions—SecureDrop, HSM-backed signing services, self-hosted transparency infrastructure—require resources, expertise, and operational capacity unavailable to the population most exposed to infrastructure-level content suppression. This gap is not a technical inevitability; it is a consequence of designing security systems for well-resourced adversaries and well-resourced defenders simultaneously.

ArchivioMD demonstrates that a complete cryptographic integrity stack—post-quantum signatures, multi-jurisdiction temporal binding, DNSSEC key corroboration, and steganographic canary tokens—can be constructed and deployed within the constraints of shared-host PHP 7.4+ without native extensions or root access. The seven-anchor trust model provides detection guarantees against a hostile infrastructure operator, legal compulsion across a single jurisdiction, and

compromised administrator credentials, while remaining verifiable by any third party with standard command-line tools and no pre-existing trust in the publisher's infrastructure.

The system's contribution is not any individual cryptographic component—FIPS 205 SLH-DSA, Rekor, RFC 3161, DANE/DNSSEC, and OpenPGP all pre-exist this work. The contribution is their composition into a system deployable on the lowest tier of commodity hosting, with a trust model explicitly designed around the constraint that the publisher does not control their own infrastructure.

A journalist running WordPress on an \$8/month shared host now has access to the same temporal binding and tamper-evidence guarantees that previously required dedicated infrastructure costing orders of magnitude more. That shift in the accessibility of cryptographic content integrity tooling is this paper's central result.

Software Artifacts:

ArchivioMD: <https://github.com/MountainViewProvisions/archiviomd>
<https://wordpress.org/plugins/archiviomd/>
 ArchivioID: <https://github.com/MountainViewProvisions/ArchivioID>
<https://wordpress.org/plugins/archivioid/>
Version 1.0 | March 2026

References

1. Bailey, S.R. (2026). ArchivioMD: Accessible Cryptographic Content Integrity for Independent Publishing Infrastructure [WordPress plugin, v1.5.9]. Mountain View Provisions LLC. <https://github.com/MountainViewProvisions/archiviomd>
2. Bailey, S.R. (2026). ArchivioID: OpenPGP Identity and Multi-Signer Verification Layer for ArchivioMD [WordPress plugin, v5.1.0]. Mountain View Provisions LLC. <https://github.com/MountainViewProvisions/ArchivioID>
3. Atallah, M.J., Raskin, V., Hempelmann, C.F., et al. (2002). Natural language watermarking and tamperproofing. In Proceedings of the 5th International Workshop on Information Hiding, LNCS 2578, pp. 196-212. Springer.
4. Bernstein, D.J., Hulsing, A., Kolbl, S., et al. (2019). The SPHINCS+ signature framework. In Proceedings of ACM CCS 2019, pp. 2129-2146. ACM.
5. Brassil, J.T., Low, S., Maxemchuk, N.F., and O'Gorman, L. (1995). Electronic marking and identification techniques to discourage document copying. IEEE Journal on Selected Areas in Communications, 13(8), 1495-1504.
6. Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and Thayer, R. (2007). OpenPGP Message Format. RFC 4880. Internet Engineering Task Force.
7. Fernandez, M., Torres-Arias, S., Diaz, C., and Kim, J. (2023). in-toto Attestation Framework. GitHub / in-toto Working Group.
8. Laurie, B., Langley, A., and Kasper, E. (2013). Certificate Transparency. RFC 6962. Internet Engineering Task Force.
9. Newman, Z., Meyers, J.S., and Torres-Arias, S. (2022). Sigstore: Software signing for everybody. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), pp. 2353-2367. ACM.
10. NIST (2023). Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters. Special Publication 800-186.
11. NIST (2024). Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standard 204.
12. NIST (2024). Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standard 205.
13. NIST (2023). Digital Signature Standard (DSS). Federal Information Processing Standard 186-5.
14. Protz, T., et al. (2023). OpenPGP.js v5. OpenPGP.js Contributors. <https://openpgpjs.org>

15. Pletcher, L., Torres-Arias, S., Meyers, B., and Newman, Z. (2021). Rekor: Immutable tamper-resistant ledger of metadata generated within a software supply chain. Sigstore / Linux Foundation.
16. Adams, C., Cain, P., Pinkas, D., and Zuccherato, R. (2001). Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161. Internet Engineering Task Force.
17. Housley, R. (2009). Cryptographic Message Syntax (CMS). RFC 5652. Internet Engineering Task Force.
18. Krawczyk, H. and Eronen, P. (2010). HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869. Internet Engineering Task Force.
19. Crocker, D., Hansen, T., and Kucherawy, M. (2011). DomainKeys Identified Mail (DKIM) Signatures. RFC 6376. Internet Engineering Task Force.
20. Topkara, M., Topkara, U., and Atallah, M.J. (2006). Words are not enough: Sentence level natural language watermarking. In Proceedings of the 4th ACM Workshop on Content Protection and Security, pp. 37-46. ACM.
21. W3C (2024). Verifiable Credential Data Integrity 1.0. W3C Candidate Recommendation. <https://www.w3.org/TR/vc-data-integrity/>
22. Wouters, P., Tschofenig, H., Callas, J., Calder, T., and Richardson, M. (2024). OpenPGP. RFC 9580. Internet Engineering Task Force.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.