



Post-Quantum Content Provenance

Headline Recorder

TECHNICAL WHITE PAPER

Muhammad Shaheer Bin Junaid

Chief Technology Officer

www.headlinerecorder.com

0

Measured figures

Figures from the production deployment on Headline Recorder. Sign times are one core, pure PHP.

NIST Cat. 5

ML-DSA-87 signatures on every article

4,627 B

ML-DSA-87 signature, served detached

~17 ms

Pure-PHP sign time, one core

8,380,417

Residues checked exhaustively

0 B

Signature overhead on a reader's page load

A

Abstract

Every article and op-ed on the site is signed with a hybrid post-quantum signature: ML-DSA-87 (FIPS 204, NIST security category 5) and ED448 (RFC 8032), over a canonical, byte-stable encoding of the stored article and its metadata. The signatures are detached and published at stable addresses. Anyone can check them in a browser, or with any conformant implementation, against a public keyset. That keyset is itself signed by an offline key, and the key identifier is pinned in DNS.

The host is commodity shared hosting. It has no native post-quantum libraries, so SHAKE, ML-DSA-87 and ED448 were written in portable PHP. The implementations match an established reference byte for byte. ML-DSA-87 was also checked against NIST ACVP known-answer vectors. The rounding and hint helpers were checked exhaustively: all 8,380,417 residues modulo q , with no mismatches. Signing sits on the publication path. Nothing is stored until it verifies. One ML-DSA-87 signature takes about 17 ms on a single core. Contributor submissions are sealed in the browser with ML-KEM-1024 + P-384, then AES-256-GCM. The server never sees the plaintext.

We have not found another news publisher that signs its full article corpus with post-quantum signatures a reader can verify. That is a statement about what we found, not a proof.

1

Motivation

Most copies of a news article are not the origin page. Screenshots, reprints, aggregator caches and archives all circulate. HTTPS authenticates a connection, not a document. Once the article leaves the origin, nothing cryptographic ties the copy to what the publisher stored. Generative tools have made convincing edits cheap. The gap is getting worse.

A signature closes that gap only for as long as the algorithm holds. RSA, ECDSA and EdDSA are expected to fall to a cryptographically relevant quantum computer. For confidentiality the usual phrase is "harvest now, decrypt later". For provenance the matching problem is "trust now, forge later": a future quantum adversary could fabricate classically signed articles that look identical to real archive entries. A news archive is meant to stay trustworthy for decades, so the signatures have to resist quantum attack from the day they are made.

Algorithm strength is not enough on its own. The signer asserts the signing time. Long-term assurance also needs independent evidence that a signature existed then: a public transparency log, or an external timestamp (section 11). NIST finalised ML-DSA (FIPS 204) and ML-KEM (FIPS 203) in August 2024. A standards-based deployment became possible after that.

2

Threat Model and Security Goals

Adversary	Capability	Goal	Countermeasure
Content forger	Publishes altered copies of articles	Make readers believe a modified text is authentic	Hybrid signature over canonical content; any change fails verification
Future quantum adversary	Breaks discrete-log and factoring-based schemes	Forge signatures on archive content	ML-DSA-87 (Module-LWE / Module-SIS)
Cryptanalytic surprise	A new classical attack on ML-DSA	Forge signatures	AND-composition with ED448: forgery requires breaking both
Key substitution	Controls a mirror or the network path	Present an attacker-made key as the publisher's	Keyset signed by offline key; kid pinned in DNS (DNSSEC where available)
Cross-protocol replay	Obtains a signature made for another purpose	Reuse it as an article signature	Context string pqsig-v1 for domain separation
Stale-signature replay	Holds an older valid signature of an edited article	Present the old version as current	Signature binds modification time and content hash; re-sign on every change
Server compromise	Full control of the web host	Obtain the signing key or submission plaintext	Offline keyset certificate; submissions sealed client-side
Network observer	Records traffic now, decrypts later	Read submissions or sessions	Hybrid PQ TLS at the edge; ML-KEM-1024 sealing end-to-end

Non-goals. Signatures attest origin and integrity, not factual accuracy. They do not hide content, and they do not prevent a compromised publisher from signing false material; revocation and a public log (section 11) address the latter.

3

Cryptographic Design

3.1 Primitives and parameters

Function	Primitive	Standard	Parameters & sizes
Post-quantum signature	ML-DSA-87	FIPS 204	$q = 8\,380\,417$; $n = 256$; $(k, l) = (8, 7)$; $\eta = 2$; $\tau = 60$; $v_1 = 2^{19}$; $v_2 = (q-1)/32$; $\omega = 75$; pk 2 592 B; sk 4 896 B; sig 4 627 B
Classical signature	ED448 (pure, context)	RFC 8032	Edwards448 over $p = 2^{448} - 2^{224} - 1$; SHAKE256; pk 57 B; sig 114 B
Content digest	SHA-512	FIPS 180-4	512-bit digest of the canonical encoding
Key identifier	SHA-256	FIPS 180-4	kid = hex(SHA-256(pk _{ML-DSA} pk _{ED448}))
Post-quantum KEM	ML-KEM-1024	FIPS 203	$(k, \eta_k, \eta_b, d_u, d_v) = (4, 2, 2, 11, 5)$; ek 1 568 B; ct 1 568 B
Classical KEM	ECDH P-384	SP 800-56A	Hybrid combiner (noble post-quantum / IETF drafts); 256-bit key
Bulk encryption	AES-256-GCM	SP 800-38D	96-bit IV; 128-bit tag; authenticated associated data
Key derivation	HKDF-SHA-256	RFC 5869	Derives KEK when no dedicated secret is configured
Password hashing	Argon2id	RFC 9106	m = 19 456 KiB, t = 2, p = 1

3.2 Hybrid composition

The signature is the pair $(\sigma_{\text{ML-DSA}}, \sigma_{\text{ED448}})$ over the same message. A document is valid only if both verify (AND-composition). An existential forgery of the hybrid is then a forgery of each component, so the construction stays unforgeable if at least one component is. The hedge covers a quantum break of ED448 and an unknown classical break of a young lattice scheme. The cost is 114 bytes and one extra verification. Both components use the context string **pqsig-v1**. ML-DSA uses the FIPS 204 pure-mode formatting $M' = 0x00 || \text{len}(\text{ctx}) || \text{ctx} || M$. ED448 uses the RFC 8032 dom4(0, ctx) prefix.

3.3 Canonical content encoding

Rendered HTML moves: ads, lazy loading, template updates. The signed object is the stored article plus the metadata a reader relies on, serialised with a fixed key order:

```
C(article) = UTF-8( JSON{
  "v":      1,
  "url":    permanent address,
  "title":  title (HTML entities decoded),
  "author":  byline,
  "published": UTC, ISO-8601,
  "modified": UTC, ISO-8601,
  "html":   stored article body
} ) with fixed key order, unescaped slashes and Unicode
```

Any edit to the text, headline, byline, dates or address changes C, and therefore the digest. C is the stored markup, not the rendered page. A valid signature proves what the publisher stored. The reader still has to compare that text with the page in front of them. The verification page shows the signed text and a side-by-side comparison for that reason.

3.4 Signature document

```
header = { "v":1, "alg":"ML-DSA-87+ED448", "kid":<64 hex>, "type":"article",
           "url":<path>, "sha512":<hex SHA-512(C)>, "signedAt":<ISO-8601 ms Z> }
H      = UTF-8(header)
document = { "protected": b64url(H),
             "mldsa87":  b64url(ML-DSA-87.Sign(sk1, H, ctx="pqsig-v1")),
             "ed448":    b64url(ED448.Sign(sk2, H, ctx="pqsig-v1")) }
```

What is signed is the protected header, not the article. Both signature operations stay the same size no matter how long the article is. The content is bound by SHA-512, and a verifier checks the exact header bytes it received, so it does not have to re-canonicalise anything. A complete document is about 6.5 KB.

3.5 Verification procedure

1. Decode **protected**; require $v = 1$ and $alg = \text{ML-DSA-87+ED448}$.
2. Establish the keyset: verify its offline signature (type **keyset**) and require the certifying key ID in the **_pqsig** DNS record.
3. Locate *kid* in the certified keyset; recompute $\text{SHA-256}(\text{pk}_{\text{ML-DSA}} \parallel \text{pk}_{\text{ED448}})$ and require equality; reject revoked keys.
4. Verify $\sigma_{\text{ML-DSA}}$ on H with context **pqsig-v1** (FIPS 204 Verify, including hint and norm checks).
5. Verify σ_{ED448} on H with the same context, with canonical-encoding checks, $S < L$, rejection of small-order keys and the cofactored equation.
6. Recompute SHA-512 of the content bytes and compare with **sha512**.

Verification never throws on hostile input; it returns a typed failure reason.

3.6 Trust anchors and key distribution

Keyset. Public keys are published as JSON at `/.well-known/pqsig-keys.json`, served as a static file with CORS enabled, mirrored through a read-only API endpoint.

DNS pin. A TXT record `_pqsig.headlinerecorder.com` carries `v=pqsig1; alg=ML-DSA-87+ED448; kid=...`. With DNSSEC the pin is authenticated independently of the web server.

Keyset certification. The keyset is signed with the publisher's offline key, which never touches the server. A compromised server cannot introduce a key of its own without the verifier showing that the key list is uncertified.

Separation of keys. The offline key certifies the keyset; a dedicated online key signs articles automatically. Either can be revoked by marking it in the keyset (followed by a new offline certificate).

Implementation on Commodity Hosting

4.1 Constraints

The platform is WordPress on shared LiteSpeed hosting, PHP 8.4. There is no FFI, no guaranteed GMP or BCMath, and no OpenSSL build with ML-DSA. PHP's OpenSSL bindings also cannot pass an ED448 context string. Signing stays on the publication path. The primitives are dependency-free PHP, using native 64-bit integers.

4.2 Keccak and SHAKE

Keccak-f[1600] operates on 25 native 64-bit lanes with the 24 rounds unrolled; SHAKE128 and SHAKE256 expose an incremental extendable-output reader so rejection sampling consumes exactly as many blocks as it needs (rates 168 and 136 bytes).

4.3 ML-DSA-87

Polynomials in $\mathbb{Z}_q[X]/(X^{256} + 1)$ are multiplied in the NTT domain using precomputed powers of the primitive 512th root of unity $\zeta = 1753$, with lazy modular reduction. Key generation expands the 8×7 matrix $\hat{A}$ from ρ with SHAKE128 rejection sampling; signing follows FIPS 204 Sign_internal with hedged randomness. Decompose, HighBits, LowBits, MakeHint, UseHint and Power2Round are written branch-free and were verified exhaustively against the specification for every residue modulo q (8 380 417 values, zero mismatches).

4.4 ED448

Field elements are 16 limbs of 28 bits. Multiplication and squaring are generated and unrolled code, with Goldilocks folding. Points use complete twisted-Edwards formulas in extended coordinates. Verification enforces canonical encodings, $S < L$, and rejects small-order keys before the cofactored check.

4.5 Validation

Test	Scope	Result
SHAKE128 / SHAKE256 against reference	30 + 30 random inputs, 0–1 400-byte outputs	Identical
ML-DSA-87 KeyGen(ξ)	30 random seeds	pk and sk byte-identical
ML-DSA-87 deterministic signing	30 cases, messages 0–1 500 B	Byte-identical
ML-DSA-87 cross-verification	60 hedged PHP ↔ reference	All accepted
ML-DSA-87 negative tests	150 tampered / malformed cases	All rejected by both
ED448 RFC 8032 §7.4 vectors	Including the context vector	Pass
ED448 against reference	30 keys + 150 cross-verification	Byte-identical; full agreement
Rounding & hint helpers	All 8 380 417 residues mod q	0 mismatches
NIST ACVP ML-DSA-87 KATs	10 keyGen + 10 sigGen + 15 sigVer	35 / 35 pass

4.6 Performance

Operation (median of 10, one core)	Interpreter	OPcache	OPcache + JIT
ML-DSA-87 sign	16–19 ms	11.9 ms	6.6 ms
ML-DSA-87 verify	8–11 ms	7.2 ms	2.7 ms
ED448 sign	32–41 ms	23.5 ms	14.7 ms
ED448 verify	18–23 ms	12.5 ms	8.9 ms

Peak memory is about 2 MB. End-to-end publishing (canonicalise, sign both, verify both, store) takes 0.2–0.3 s; the backlog signer sustains about 100 articles per 8-second batch.

4.7 Side-channel posture

Interpreted PHP is not constant time. We do not claim that it is. Secret-dependent branches and table indices are avoided where that is practical. That reduces leakage under an interpreter; it does not remove it. Intermediates are wiped where the runtime allows. Signing is confined to authenticated publication and scheduled jobs, on content the publisher chose, so an attacker does not get an adaptive timing oracle. Hedged ML-DSA randomness also cuts exposure to fault attacks and to deterministic-nonce mistakes.

5 Key Management

Envelope encryption. The online key is stored as AES-256-GCM ciphertext with a fresh 96-bit IV and a mandatory 128-bit tag. Associated data binds a format tag, the key ID, the site's canonical address, the key's source and the KEK type.

Key-encryption key. A dedicated 256-bit secret held in server configuration, or derived with HKDF-SHA-256 from installation secrets. The key file lives outside the public web root with owner-only permissions.

Import validation. An imported key must match a non-revoked keyset entry: ρ and tr agree, the derived ED448 public key matches, and a sign-and-verify round trip succeeds before anything is stored.

Operator controls. Every key operation requires the administrator's password and second factor, a same-origin request token, passes a rate limiter and is written to an audit log. A known-answer self-test runs once per runtime before the key is used.

6 Signing Pipeline

Triggers. Publication and every update are signed inside the same request, under a time budget. A scheduled sweep every five minutes signs anything pending or stale, newest first, within a 20-second budget.

Concurrency. An expiring lock with atomic refresh serialises the sweep, the operator backlog and command-line runs, so two writers never race on the same article.

Verify-before-store. Every signature is verified with the server's own ML-DSA-87 and ED448 verifiers before it is stored. Uploads that are stale, for another article, from an unknown key or malformed are refused.

Storage & delivery. Signature documents live in a dedicated table keyed by post. They are served at `/signed/<id>.sig.json` with permissive CORS, advertised from each article with `<link rel="alternate">`, and purged from the edge cache whenever they change. RSS and podcast feeds are snapshotted and signed as files.

7 Sealed Contributor Submissions

```
header = { "v":1, "alg":"ML-KEM-1024+P-384/AES-256-GCM", "kid":<editor key id> }
(ct_kem, ss) = HybridKEM.Encaps(ek_editor)          # ML-KEM-1024 + P-384
iv       = random(12)
ct       = AES-256-GCM.Enc(ss, iv, plaintext, aad = header || ct_kem)
envelope = { protected: b64url(header), kem: b64url(ct_kem), iv, ct }
```

Encapsulation and encryption run in the contributor's browser; the server stores only the envelope and minimal routing metadata. The decapsulation key is held offline by editors. Binding the header and KEM ciphertext as associated data prevents envelope splicing and key-ID substitution. Images are re-encoded before sealing, removing EXIF and GPS metadata.

8 Transport and Platform Security

Transport. TLS 1.3 at the edge with hybrid post-quantum key agreement (X25519 + ML-KEM-768) for supporting clients, HTTP Strict Transport Security, and a tightened content security policy on account pages.

Authentication. WebAuthn passkeys, TOTP with replay protection and single-use recovery codes, Argon2id password hashing, breached-password screening by k-anonymity range queries, uniform error responses, and per-IP / per-account throttling.

Request integrity. Capability checks, request tokens and same-origin validation on every state-changing endpoint; prepared database statements throughout.

Edge cache safety. Shared cache headers are issued only for anonymous, cookie-free responses; any cookie or header that could influence markup is excluded from shared responses.

Assessment. A grey-box penetration test covered authentication, registration, second-factor flows, CSRF, object-level authorisation, XSS, SQL injection, SSRF, path traversal, upload handling, cache poisoning, CORS and information disclosure. No high or critical finding was reproduced. Every module was then audited line by line.

9

Engineering Challenges

Post-quantum signing without native libraries

Portable PHP implementations of SHAKE, ML-DSA-87 and ED448, validated byte-for-byte and exhaustively, meeting latency targets by more than 12×.

ED448 with a context string

PHP's OpenSSL bindings cannot pass a dom4 context, so ED448 with domain separation was written directly. Base-point multiplication uses a fixed operation sequence.

Authenticated-encryption edge cases

Tags are now fixed at 128 bits and the associated data uses the stored canonical address after review found shortened tags would have been accepted.

Concurrent signers

An expiring lock with atomic refresh coordinates the publication hook, scheduled sweep, operator backlog and command-line signer.

Signature data on the hot path

Moving 6.5 KB signature documents into a dedicated table removed them from every archive query without changing any public interface.

Key discovery on managed hosting

Hosting control panels intercept /.well-known/; the keyset is written as a static file and mirrored through an API.

Integrity of journalists' text

House-style rewriting is limited to site-owned text. A reporter's article is not rewritten by software, so the signature stays stable.

10

Related Work and Positioning

System	Object signed	Post-quantum	Independent verification
Conventional news sites	None (transport only)	No	No
Post-quantum TLS (CDN default)	Connection	Key agreement	Only in transit
C2PA Content Credentials	Mainly images & video manifests	ML-DSA planned	Yes
Apple Reference Image (reported)	Images	RSA-3072 + ML-DSA-87	Within vendor ecosystem
Static-site provenance proposals	Build artefacts	Proposed	Not deployed
Headline Recorder	Full article corpus and feeds	ML-DSA-87 + ED448	Any browser or library

A search of public sources found no other news publisher signing its corpus with post-quantum signatures. That is what the search found. It is not a proof. Corrections are welcome.

11

Limitations and Future Work

- **Audit.** The PHP primitives are validated against a reference but have not had an independent cryptographic audit; we intend to commission one.
- **Online key.** Automatic signing requires a key on the server. A hardware-backed or remote signer is planned.
- **Transparency and time.** A publicly auditable append-only log of signatures (Certificate Transparency style), or RFC 3161 timestamps, would provide independent evidence of existence time and enable revocation with a validity window.
- **Visible-text binding.** A browser extension that extracts the article text from the page and compares it with the signed content automatically.
- **Interoperability.** A C2PA-compatible manifest once ML-DSA enters that standard, and a browser extension that verifies articles and syndicated copies automatically.

- **Specification.** Publishing the canonical encoding and envelope format as an open specification so other publishers can adopt it.

12 Reproducibility

1. Fetch any article's signature from <https://www.headlinerecorder.com/signed/<id>.sig.json> (linked from the article's head) and the keyset from [/.well-known/pqsig-keys.json](https://www.headlinerecorder.com/.well-known/pqsig-keys.json).
2. Fetch the keyset certificate from [/wp-json/hr/v1/signing/keyset-certificate](https://www.headlinerecorder.com/wp-json/hr/v1/signing/keyset-certificate), verify it, and confirm the certifying key ID with `dig +dnssec TXT _pqsig.headlinerecorder.com`.
3. Fetch the canonical bytes from the public canonical-content endpoint and verify per section 3.5 with any FIPS 204 and RFC 8032 implementation, using context `pqsig-v1`. The browser verifier is at www.headlinerecorder.com/verify.

12.1 Live check (24 October 2026)

Checked against the live site. Nothing staged for this paper.

- Keyset: www.headlinerecorder.com/.well-known/pqsig-keys.json
- Two keys published, both `ML-DSA-87+Ed448`.
- Active `kid` on recent articles: `b63130ccf2ed53987bf2fcb2e10975e77add04b25f41762236d98d306a83264`
- Example article: post 6117, [/2026/10/22/audi-appoints-former-chery-executive/](https://www.headlinerecorder.com/2026/10/22/audi-appoints-former-chery-executive/)
- Signature file: [headlinerecorder.com/signed/6117.sig.json](https://www.headlinerecorder.com/signed/6117.sig.json)
- Linked from the article head as `rel="alternate" type="application/json" title="Post-quantum signature"`
- Envelope: `v=1,type=article,signedAt=2026-10-23T22:24:11.204Z`
- Detached `mldsa87` and `ed448` fields both present.
- Keyset certificate: [/wp-json/hr/v1/signing/keyset-certificate](https://www.headlinerecorder.com/wp-json/hr/v1/signing/keyset-certificate)
- Browser verifier: www.headlinerecorder.com/verify

A Appendix: example keyset entry

```
{ "keys": [ {  
  "alg": "ML-DSA-87+Ed448",  
  "kid": "b63130ccf2ed53987bf2fcb2e10975e77add04b25f41762236d98d306a83264",  
  "mldsa87": "<base64url, 2,592 bytes>",  
  "ed448": "<base64url, 57 bytes>"  
} ] }
```

B Appendix: platform context

Signing is one piece of a wider release: reader accounts with passkeys and two-factor authentication, a reading-habit tracker, a daily audio briefing with a standards-compliant podcast feed, edge delivery with cache purge on publication, structured data and news sitemaps, and a machine-readable summary for answer engines. Those parts use the controls in section 8.

HEADLINE RECORDER

contact@headlinerecorder.com · www.headlinerecorder.com

Muhammad Shaheer Bin Junaid

Chief Technology Officer, Headline Recorder

The code was reviewed and tested before release.

Standards: FIPS 203 · FIPS 204 · FIPS 180-4 · SP 800-38D · RFC 8032 · RFC 5869 · RFC 9106