Zegel

A plain-language explainer

Tamper-evident proof that a file existed, unaltered, at a moment in time. Sealed with hybrid post-quantum signatures, witnessed in a transparent log, and verifiable offline by anyone — without needing to trust us.

v0 working draft

For grant reviewers, design-partner SMBs, auditors, lawyers, and the curious

Contents

1 The thirty-second pitch

Zegel produces tamper-evident proof that a file existed, unaltered, at a specific moment in time. The proof is small, portable, and verifiable offline by anyone with the file and the receipt — including auditors, regulators, insurers, customers, and courts.

We sell that proof as ready-made compliance evidence to small and medium-sized Dutch businesses (SMBs) under the new *Cyberbeveiligingswet* (NIS2). Today, when a Dutch SMB is asked *"are these your original files?"* after a ransomware attack, a tax audit, or a supply-chain incident, the honest answer is usually *"we believe so."* Zegel turns that answer into a thirty-second cryptographic check.

The three guarantees, in one sentence each

- **Detection.** If a sealed file changes by even one byte, the verifier will say so.
- **Pre-attack proof.** Even after a breach, the recorded fingerprints of files *before* the breach remain valid and verifiable.
- **No vendor trust required.** Anyone who has the file and the receipt can verify offline, without our servers being online and without trusting us at all.

Three properties that make this work

- **Hybrid post-quantum signatures.** Every entry is signed twice: once with classical Ed25519, and once with ML-DSA-65, NIST's post-quantum standard. A future quantum computer that breaks one still cannot forge entries — both must fall.
- **Witnessed transparent log.** Every entry is added to an append-only ledger whose root is co-signed each hour by independent Dutch witnesses (SURF, NLnet Labs, a notary). No single party — including Zegel itself — can rewrite history.
- **EU-sovereign by construction.** Hosted on Dutch infrastructure, operated under Dutch jurisdiction, with Dutch-language NIS2 reporting out of the box.

Who this document is for

This explainer is written so that a careful non-programmer can finish it and understand exactly what Zegel does, why it matters, and what would have to be true for it to fail. We define every technical term on first use, ground every claim in a real-world scenario, and call out the limits of the system with the same energy we describe its strengths.

Specifically, it is written for grant reviewers (NLnet, the Sovereign Tech Fund, RVO MIT-haalbaarheid), early design-partner SMBs, journalists writing about Dutch cyber-resilience, and any auditor or lawyer reviewing the system on behalf of a buyer.

> **WHAT YOU DO NOT NEED TO KNOW TO READ THIS**
>
> You do not need to be a programmer or a cryptographer. We will not ask you to compute a hash by hand. We will, however, ask you to trust pictures and analogies more than equations, and to read carefully when we mark something as a *limit*. Limits are where bad pitches hide; we put ours up front.

2 The problem we are solving

When something goes wrong with the files on a small business's servers, someone always ends up asking the same question: *"are these the originals, unmodified?"* Today, the honest answer is usually *"we believe so."* That answer is fine for a bookkeeper's daughter; it is not fine for an insurer writing a six-figure cheque, an auditor opening a NIS2 file, or a lawyer preparing for litigation.

We call this the **integrity gap**. The files exist. The question of whether they are the same files that existed last year also exists. What does *not* exist — for the vast majority of Dutch small businesses — is a quick, convincing procedure to answer the question without anyone having to take anyone's word for it.

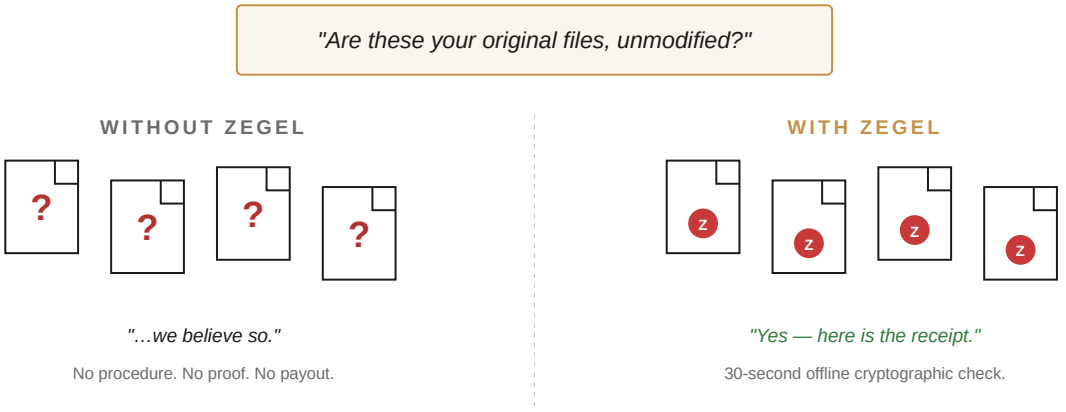


Diagram 1 — The integrity gap, before and after.

Three places the integrity gap is felt

The same primitive — a tamper-evident, witnessed, post-quantum record of file fingerprints — solves three apparently separate problems. Each one is normal, each one is expensive, and each one ends with somebody being unable to answer the question above.

1. Ransomware recovery

About **fifty thousand Dutch SMBs** are hit by ransomware each year, according to figures used by the Ministry of Economic Affairs. After an attack, the IT team usually has backups — but the painful question is which version of which file is the genuine pre-attack state. In the 2019 attack on Maastricht University, where a €197 000 ransom was eventually paid in Bitcoin, the longest single phase of recovery was not the technical restoration but the validation of which restored files could be trusted. With Zegel, that validation collapses from days of forensic work to a per-file, thirty-second cryptographic check. Insurers increasingly require proof of pre-attack state before paying out claims; Zegel produces it out of the box.

2. Supply-chain integrity

Under NIS2 Article 21(d), so-called *essential and important entities* — hospitals, gemeenten, large retailers, energy operators — are obligated to verify the integrity of their suppliers. In practice, this is impossible today: a hospital cannot meaningfully audit two hundred small suppliers' file systems on demand. With Zegel, an anchor organisation can mandate that its suppliers seal critical files daily, and then verify any of them later, independently, without trusting a single supplier statement. This is the *anchor-organisation sales motion* — one signed contract pulls dozens of compliance-grade suppliers into the system at once.

3. Silent historical revision

The rarest but most damaging of the three modes is the insider who quietly modifies historical records to hide a problem. Volkswagen's Dieselgate, the Wells Fargo unauthorised-accounts scandal, and the original Dutch Diginotar incident are all cousins of this failure: changes were made to records that should have been immutable, and the cover-up persisted for months or years before discovery. With Zegel, every record's hash is co-signed each hour by independent witnesses; a retroactive edit is detectable to anyone who checks, including the auditor herself.

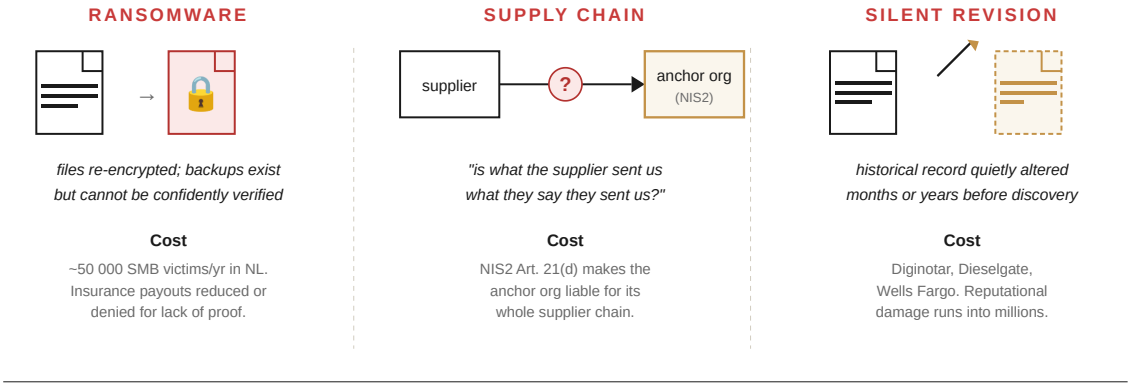


Diagram 2 — Three failure modes, one underlying gap.

The legal pressure: from optional to obligatory

Until very recently, Dutch SMBs could treat file integrity as a best-effort concern. That changed in October 2024 when the **Cyberbeveiligingswet** — the Dutch transposition of the EU NIS2 Directive — entered into force. Article 21 of NIS2 imposes ten specific risk-management measures on essential and important entities, including: *policies on the use of cryptography*, *integrity verification of network and information systems*, *supply-chain security*, and *regular effectiveness testing of cybersecurity measures*.

Zegel maps directly onto five of the ten measures (a, c, d, f, and h in the architecture document). It is, by construction, an *effectiveness check* rather than a control: every sealed file is itself the audit evidence that the integrity control is working. That is unusual, and it is the reason auditors react well to the design.

CONCRETE LEGAL EXPOSURE

Under the Cyberbeveiligingswet, regulators can issue administrative fines of up to **€10 million or 2% of global turnover** (whichever is higher) for serious non-compliance. SMBs designated as *important entities* are not exempt. Insurance underwriters are repricing accordingly. The window in which "we believe so" is acceptable is closing fast.

3 How Zegel works

Three things happen, in this order, every time a file gets sealed: a **file** on the customer's disk is observed by an **agent**, the agent computes a fingerprint and signs it twice, and the signed fingerprint is appended to a **transparent log** watched by independent witnesses. Each piece is small. The interesting property is the way they fit together.

The three actors

Zegel deliberately separates concerns into three independent parties so that no single one of them needs to be fully trusted. The agent does not trust the log; the log does not trust the agent; and the verifier — anyone holding a receipt later — trusts neither, because the receipt itself is the proof.

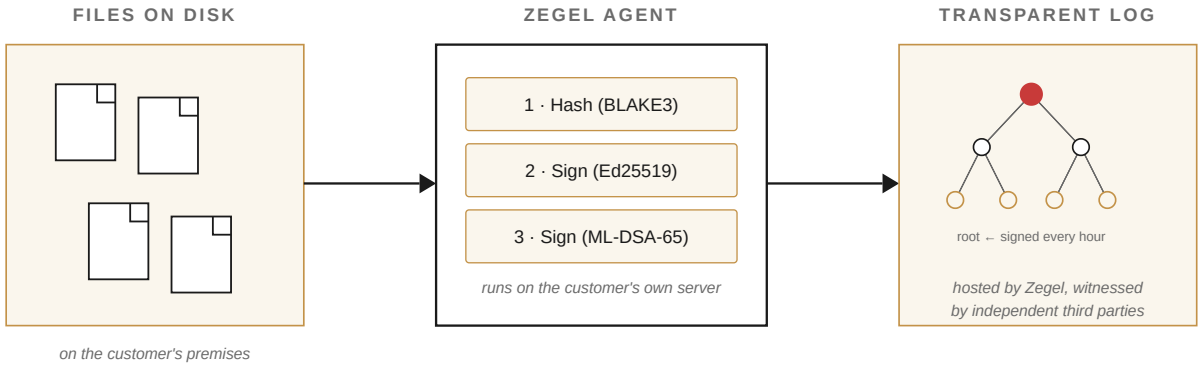


Diagram 3 — The three actors. None fully trusts the others; the receipt is the proof.

Step one: hashing — turning a file into a fingerprint

A **hash function** is a one-way procedure that turns any file — of any size, in any format — into a small, fixed-length string of characters. We use **BLAKE3**, a modern hash that is, somewhat amazingly, faster than copying the bytes themselves on a current laptop. The output is 32 bytes, normally written as 64 hexadecimal characters. Change even one byte anywhere in the file and the hash changes completely.

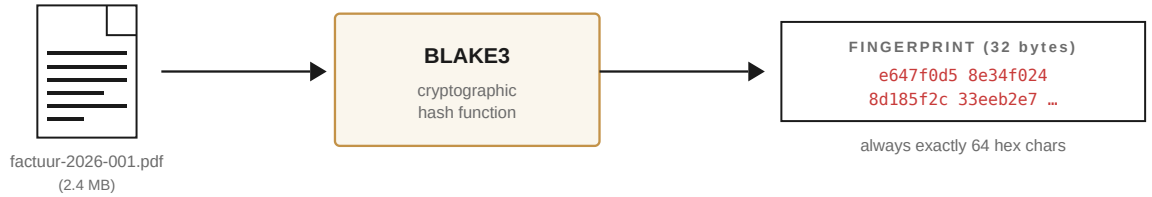


Diagram 4 — The hash function. Many-to-one, one-way, collision-resistant.

The fingerprint is short (256 bits), unique in any practical sense, and computed without any secret keys — anyone with the file can recompute it. That is exactly the property a **verifier** later wants: hand them the file and the receipt, and they can independently re-derive the fingerprint and check it matches what was recorded.

> **WHY A HASH AND NOT THE FILE ITSELF?**
>
> Because *the file never leaves the customer's premises*. That is the privacy promise. Only the fingerprint is sent to Zegel. From the fingerprint we cannot reconstruct the file contents — that is the entire point of a one-way hash. A Zegel database breach therefore exposes which customers had which agents running when, but not what was in the files.

Step two: signing — and why we sign twice

A **digital signature** is the cryptographic equivalent of a wax seal. The signer holds a *private key*; the world holds the matching *public key*. With the private key the signer can produce a signature over a message; with the public key anyone can check that this signature came from the holder of the private key. There is no known way to produce a valid signature without the private key.

Zegel signs every entry **twice**, with two different schemes:

- **Ed25519** — a classical signature scheme based on elliptic curves, in deployment for over a decade in TLS, SSH, GitHub, and every major operating system. Fast, small (64-byte signatures), well understood. Not safe against a future large-scale quantum computer.
- **ML-DSA-65** — the post-quantum signature scheme standardised by NIST as FIPS 204 in August 2024. Larger signatures (~3.3 kB), more recent, designed specifically to withstand quantum attacks based on Shor's algorithm.

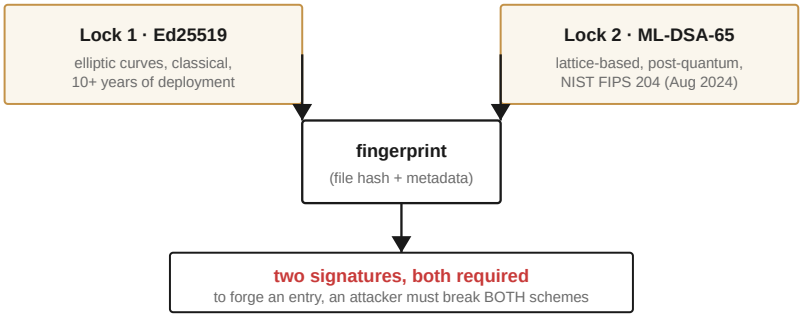


Diagram 5 — Hybrid signing. Two locks made by different lock-makers, both on the same door.

The reason for using both rather than just the post-quantum one: ML-DSA is new. It will eventually become as scrutinised as Ed25519, but it is not there yet. Rather than betting on a single horse during the transition, we sign with both. An attacker has to break *both* to forge an entry; a breakthrough against either still leaves

the other holding the door.

This is precisely the recommendation of NIST, ENISA, and the Dutch *Nationaal Cyber Security Centrum* for transition- era cryptography. It is what the Dutch financial sector is already migrating its signing infrastructure to. By doing it now, Zegel produces receipts that will still be valid in 2040.

3 How Zegel works — *continued*

Step three: appending to the transparent log

Once the agent has hashed the file and signed the result with both keys, it ships the signed bundle to the **transparent log**. A transparent log is an append-only ledger with two cryptographically enforced properties:

1. **Append-only.** Once an entry is added, its position cannot change. The log software is mathematically prevented from rewriting earlier entries without changing the log root that everyone has already seen.
2. **Inclusion provable.** For any entry, the log produces a small *inclusion proof* — typically a handful of hashes — that proves the entry is in the log without anyone having to download the whole log.

The data structure that makes this work is called a **Merkle tree**, after Ralph Merkle who invented it in 1979. Each leaf is the hash of one log entry; each internal node is the hash of its two children; the root is a single 32-byte hash that summarises everything in the log. Knowing the root pins down the entire history.

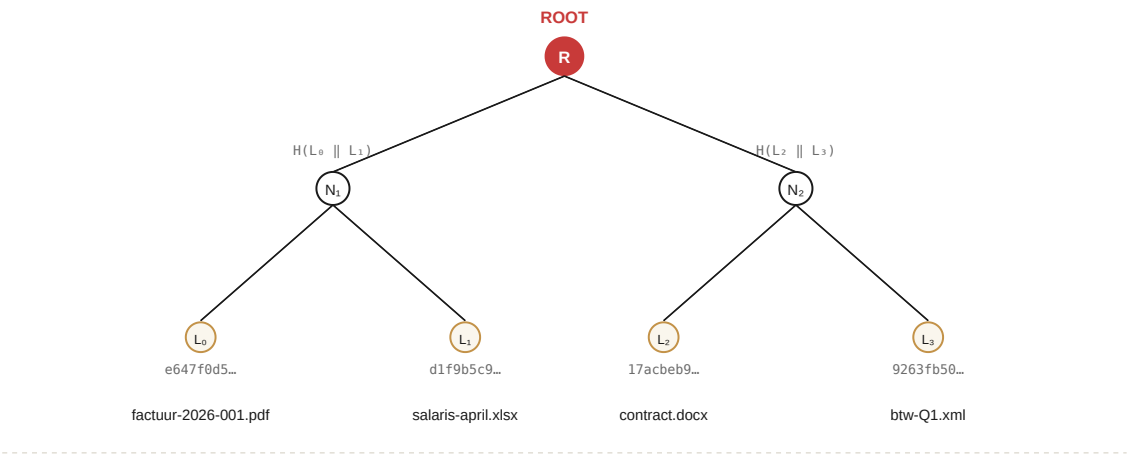


Diagram 6 — A four-leaf Merkle tree. The root is a 32-byte summary of the whole log.

In practice the log holds millions of entries, not four; the tree is wider but the principle is identical. A receipt for any single entry contains an inclusion proof — a path of just $log_2(N)$ hashes from leaf to root — and the verifier can replay that path to confirm the entry was present under the root.

Step four: witnessing — and why it protects you against us

Zegel hosts the log. What stops Zegel from secretly running two logs — one for honest customers, another with forged history for someone who pays us off? The answer is the **witness federation**.

Each hour, Zegel publishes the current log root and asks a small group of independent organisations to co-sign it: SURF (the Dutch academic backbone), NLnet Labs (operators of national-importance DNS and RPKI), a Dutch notary, and eventually a sectoral *brancheorganisatie* for the customer's industry. Each witness sees only the 32-byte root hash — never the entries, never the file metadata, never the file contents. Their job is solely to put their independent signature on the statement *"I saw root R at time T."*

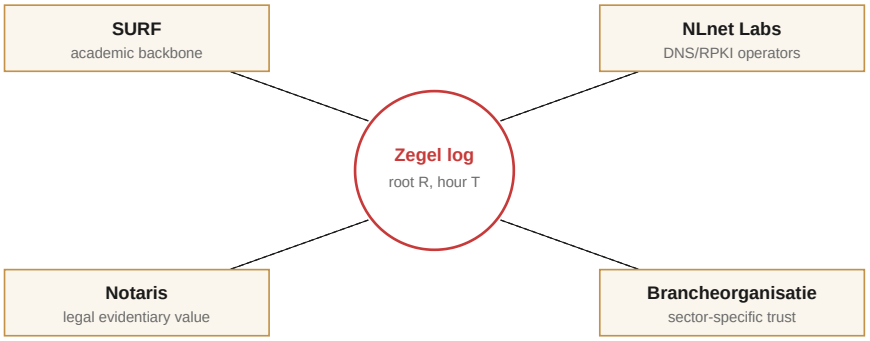


Diagram 7 — The witness federation. Each witness independently co-signs the log root every hour.

The trick: if Zegel ever serves two different roots — one to honest customers, another to the colluding party — the witness signatures on those roots will not all agree. Any auditor comparing two customers' receipts can detect the fork mathematically. This is called **split-view detection** and it is the property that justifies the whole open-source- agent + hosted-control-plane model: the customer does not need to trust Zegel; they need to trust the witness federation collectively, and we engineer that to be a bar Zegel itself cannot clear.

> **WHY THIS IS STRONGER THAN A BLOCKCHAIN**
>
> A permissioned blockchain (Hyperledger Fabric, the original Privachain proposal) gives you the same tamper-evidence property at much higher operational cost. A public blockchain like Ethereum gives you witnessing-by-the-world but adds a coin, a consensus mechanism, and dependence on non-EU validators. The witness federation gets the tamper-evidence of a blockchain at *~1% of the operational cost*, with EU jurisdiction, no token, and no consensus risk.

Step five: verifying — the magic of "no internet required"

The whole design exists so that an auditor, six months after the fact, can take only the file on disk and the receipt JSON next to it, and prove the file's authenticity offline. No call to Zegel's servers. No call to the witnesses. Just maths.

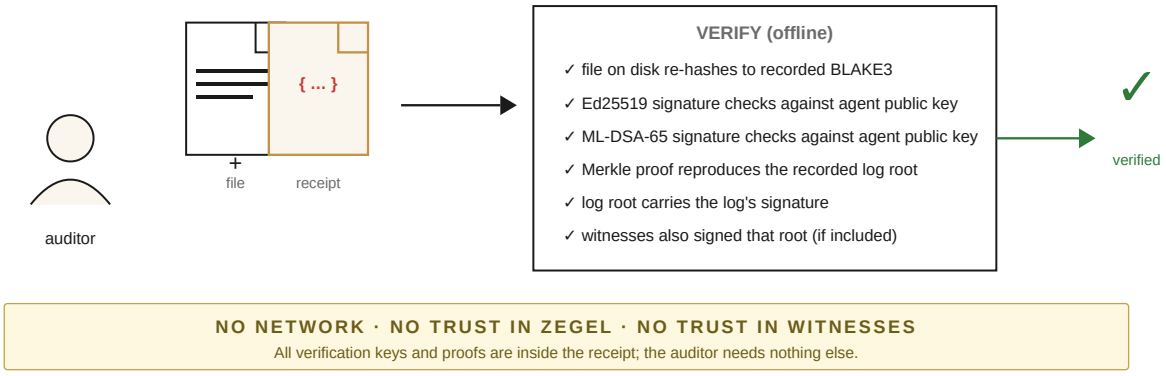


Diagram 8 — Offline verification. The receipt is self-contained.

The verifier walks four checks: re-hash the file, verify both signatures, replay the inclusion proof up to the root, and verify the root signatures (the log's, and the witnesses' if present). If all checks pass, the receipt is valid. If any check fails, the verifier reports precisely which one failed and why — content tampered, signature invalid, log root forged, and so on.

This is the property that makes Zegel useful *even if Zegel itself disappears tomorrow*. The receipts are self-contained. The auditor needs nothing from us, ever again. That property is also our strongest defence against being held hostage by future commercial pressure: a customer who does not like our pricing can walk away with all of their evidence intact.

4 Glossary of terms

Every technical term used in this document, defined briefly. If a definition uses other terms, they are defined elsewhere on this page. The glossary is alphabetical for reference; you do not have to read it in order.

Agent (`zegel-agent`)

The small program that runs on the customer's own server, watches files, computes their fingerprints, signs them, and submits the signed fingerprints to the log. Open source. Generates its own keys locally; the keys never leave the customer's premises.

BLAKE3

The hash function we use. Modern, fast (faster than copying bytes), and well-reviewed. Output is 32 bytes (256 bits), normally written as 64 hexadecimal characters.

Cyberbeveiligingswet

The Dutch law transposing the EU NIS2 Directive. In force since October 2024. Imposes ten cybersecurity risk-management measures on essential and important entities.

Ed25519

A classical (non-post-quantum) digital signature scheme based on elliptic curves. Fast, widely deployed, well understood. One half of our hybrid signature.

Fingerprint

Informal name for a hash. See *hash*.

Hash

A short, fixed-length, one-way summary of any data. Two identical files produce identical hashes; two different files produce different hashes (with overwhelming probability). From a hash you cannot reconstruct the original file.

Hybrid signature

Two independent digital signatures over the same message, one classical (Ed25519) and one post-quantum (ML-DSA-65). To forge an entry an attacker must break both schemes. The recommended NIST/ENISA pattern for transition-era cryptography.

Inclusion proof

A small piece of data (a few hashes) that proves a specific entry was included in the log under a specific log root, without requiring the verifier to download the whole log.

Ingest API

The HTTPS endpoint, hosted by Zegel, that receives signed entries from agents and appends them to the transparent log. Returns an SCT.

KvK

The Dutch chamber of commerce (*Kamer van Koophandel*). Registering a one-person business (ZZP) costs €80 and takes an afternoon.

Leaf

A single entry in the transparent log, identified by its position (an index 0, 1, 2, …) and its leaf hash.

Log root

The single 32-byte hash at the top of the Merkle tree. It summarises the entire log up to a moment in time. If the log changes, the root changes.

Merkle tree

A tree-shaped data structure (after Ralph Merkle, 1979) in which each leaf is a hash and each internal node is the hash of its two children. The root is one hash that fixes everything below it.

ML-DSA-65

A post-quantum digital signature scheme standardised by NIST as *FIPS 204* in August 2024. Lattice-based. Larger signatures than Ed25519, but resistant to attacks by quantum computers. The other half of our hybrid signature.

NIS2

The EU's *Network and Information Security Directive 2* (Directive 2022/2555). Imposes obligatory cybersecurity measures on essential and important entities across the EU. Transposed into Dutch law as the Cyberbeveiligingswet.

NLnet Labs

A Dutch non-profit operating internet infrastructure of national importance (DNS, RPKI). Likely witness federation member.

Notaris

A Dutch civil-law notary, member of the *Koninklijke Notariële Beroepsorganisatie*. Likely witness federation member; their participation gives the log legal evidentiary value in Dutch courts.

Post-quantum cryptography

Cryptography designed to remain secure against attackers with a future large-scale quantum computer. NIST standardised the first set of post-quantum algorithms in August 2024.

Private key / public key

A cryptographic key pair. Private kept secret by its owner; public shared with the world. The signer uses the private key to sign; anyone can use the matching public key to verify.

Receipt

The JSON file that `zegel-agent` writes next to a sealed file. Contains the signed entry, the SCT, the inclusion proof, and the log's public key. With the receipt and the file, anyone can verify offline.

SCT (Signed Certificate Timestamp)

The log's signed promise that an entry was included at a particular position at a particular time. Borrowed from RFC 6962 (Certificate Transparency).

Split-view attack

An attack in which a log operator shows different log roots to different parties. Detected by the witness federation: independent witnesses cannot all sign incompatible roots.

SURF

The Dutch academic and research network (*Stichting SURF*). Operator of the academic backbone. Likely witness federation member, neutral and technically credible.

Tamper-evidence

The property that any unauthorised modification to a record is detectable to anyone who checks. Stronger than tamper-resistance (which would require the modification to be prevented).

Transparent log

An append-only ledger with cryptographic guarantees of consistency and inclusion. Pioneered by Google's Certificate Transparency project; standardised in RFC 6962.

Trillian

Google's open-source implementation of a transparent log. Apache 2.0 licensed. Used by Certificate Transparency, the Go module proxy, and Sigstore. The eventual production backend for the Zegel ingest.

Verifier (`zegel-verify`)

The CLI tool an auditor runs to check a receipt against the file. Open source, ~500 lines of Rust, no network access required.

Witness federation

A group of independent organisations (SURF, NLnet Labs, a notaris, a brancheorganisatie) that each independently co-sign the log root every hour. Defends against a Zegel operator that tries to forge or rewrite the log.

ZZP

Dutch *zelfstandige zonder personeel* — a registered one-person business. The lightest legal vehicle in NL. Sufficient for invoicing customers and applying to most national grants.

5 What Zegel is not

Bad pitches hide their limits. We put ours up front, in their own section, before the comparison and the unique-selling-point pages. If a customer or a grant reviewer reads only this page and section 1, they should still understand exactly what they are buying — and what they still need to buy from someone else.

Zegel is not a backup tool

The agent never stores or transmits the contents of files. Only fingerprints (hashes) and metadata leave the customer's premises. That is by design — it is the privacy promise — but it also means Zegel cannot give your files back after a ransomware attack, a disk failure, or accidental deletion. Pair Zegel with a real backup (Veeam, Borg, Restic, Synology, a separate off-site disk). Backups give you the file back; Zegel proves what came back is genuine.

Zegel is not antivirus, EDR, or a SOC

Zegel observes file changes; it does not block them, quarantine malware, or detect intrusions on the network layer. A ransomware attack will still complete; what Zegel changes is the time-to-detect (minutes instead of days) and the quality of the post-incident proof. Pair with Wazuh, Microsoft Defender, CrowdStrike, or another endpoint detection product. We explicitly do not compete with them.

Zegel is not a blockchain

The original *Privachain* deck used the word *blockchain*. We do not, and the architecture document explains in detail why we dropped it. Zegel has no token, no consensus mechanism, no validator staking, no public ledger with everybody's data on it. What it has is a tamper-evident append-only log — the same primitive Certificate Transparency uses to keep the SSL/TLS ecosystem honest, and the same primitive Sigstore uses to keep open-source supply chains honest. Auditors recognise this primitive. Customers do not have to learn what a blockchain is.

Zegel is not a complete NIS2 platform

NIS2 Article 21 lists ten cybersecurity measures. Zegel is the *integrity* and *effectiveness-checking* leg (clauses a, c, d, f, h). The other legs — vulnerability management, training, business continuity, MFA enforcement, HR security — are out of scope. We integrate cleanly with products that handle them; we do not try to replace them.

Zegel cannot detect attacks before the agent ever ran

The agent must be running, and the agent's machine must not already be compromised at install time. If an attacker replaces a file before Zegel ever hashes it, Zegel will faithfully record the (already tampered) file as the "original." This is why the agent's *baseline scan* at install time matters legally: it pins down the *"as recorded on date T"* state, and the customer's legal defence rests on date T being plausibly clean.

Zegel cannot replace witnesses with itself

Without an external witness federation, Zegel could in principle rewrite history if its operator is compromised or coerced. Tamper-evidence becomes meaningful only when the log root is co-signed by independent third parties. Until the witness federation is operational (target: NLnet Labs and SURF by end of phase 2 in the architecture document), customers should treat Zegel as a single-vendor system with the trust profile of a single vendor.

> **THE HONEST ONE-LINE SUMMARY**
>
> Zegel does not prevent attacks, and it does not recover from them. It makes attacks *visible*, makes the recovery *provable*, and makes the audit conversation that follows *cryptographic instead of conversational*. Sold alongside a real backup and a real EDR, it closes a gap that nothing else on the Dutch SMB market currently fills.

6 How we compare to what already exists

Nothing in Zegel is unprecedented. Every cryptographic primitive we use has been deployed in production for years; the data structures we use are standardised; the witness federation idea is borrowed from RFC 6962 (Certificate Transparency) and the Sigstore project. What is new is the *combination*, and the audience: nobody else is bundling these pieces together for a Dutch SMB to install in thirty minutes and produce an auditor-ready report by day two.

This section walks through six categories of existing tooling and explains, fairly, where each one stops and where Zegel starts.

Wazuh, OSSEC, Tripwire, AIDE — file integrity monitoring

These are the classical **file integrity monitoring** (FIM) tools. They hash files on disk, store the hashes locally, and alert when a hash changes. Wazuh is the modern open-source standard and ships with a serious dashboard; AIDE is a single-binary Unix workhorse; Tripwire is the historical commercial leader.

What they do well: detecting that something on disk changed, and surfacing the change in real time. What they do not do: produce *portable*, *third-party-witnessed*, *post-quantum-signed* proof of pre-attack state. A Wazuh database is a database on the customer's own server; if the attacker compromises the server, they compromise the audit trail. The audit trail itself is therefore not court- grade evidence on its own. Zegel is best understood as *FIM with a witnessed external log* — the same first step, plus the tamper-evidence guarantee.

Sigstore / Rekor — the closest cousin in spirit

Sigstore (originally a Linux Foundation project, used by npm, PyPI, Kubernetes and the Go module proxy) is the closest thing to Zegel in cryptographic shape. It is a tamper-evident transparency log (called *Rekor*) built to record signed attestations about software artefacts. Same Merkle tree, same witnessing pattern, same public-good positioning. Where it differs is the audience: Sigstore secures the *software supply chain*, helping developers prove that a binary they shipped came from the source they claim. It does not target SMB document integrity, has no Dutch-language NIS2 onboarding, and is not designed for non-technical operators. Zegel is, in effect, "Sigstore's primitive applied to the SMB document supply chain, with Dutch jurisdiction and audit packaging."

Trillian + Certificate Transparency — the underlying primitive

Trillian is Google's open-source implementation of a transparent log; Certificate Transparency (CT) is the production deployment of Trillian that keeps the SSL/TLS ecosystem honest. CT is the reason a fraudulent SSL certificate cannot stay hidden for long: every issued certificate has to land in a public log, witnessed by

independent operators. Zegel's eventual production backend uses Trillian directly, with the same RFC 6962 hashing, because it is the most production-tested transparency log on Earth. CT itself is not a product — it is infrastructure for a specific protocol (TLS). Zegel is the productisation of the same primitive for a different document class.

Hyperledger Fabric — what the original Privachain deck used

Fabric is a permissioned blockchain framework, popular in enterprise consortia for asset-tracking and shared-ledger scenarios. The original Privachain deck proposed it as the tamper-evidence layer. We dropped it for three reasons. First, it adds a consensus protocol that the SMB neither needs nor understands. Second, it adds operational complexity (channels, orderers, peers) that costs roughly two orders of magnitude more than a transparent log to run. Third, auditors do not recognise the resulting evidence as different in kind from a witnessed Merkle log — they get the same tamper-evidence property either way. Hyperledger Fabric is the right answer for a thirty-bank consortium asset-tracking project; it is the wrong answer for a hundred-employee accountancy in Eindhoven.

Veeam, Borg, Restic, Synology — backups, not integrity proof

These are excellent products. They keep copies of files. They do not, by themselves, prove that what they hand back to you after an incident is the genuine pre-incident file rather than a quietly tampered version. (Some of them sign their backup chains, which is good; but that signature is by the same vendor whose system was potentially compromised — not by independent third parties.) We complement these tools rather than competing with them. The recommended SMB stack is *backup + Zegel + EDR*: backup gives you the file back, Zegel proves what came back is genuine, EDR catches the attacker on the way in.

Sectigo, DocuSign, Aangetekend Mailen — adjacent but different

These are document-signing services. They prove that *this document was signed by this party*, useful for contracts and notarised mailings. They do not produce verifiable evidence about *files on a customer's server changing or not changing over time*. Different problem, different solution. There is no overlap in audience or compliance use-case.

The comparison matrix

The table below summarises which capability each tool family provides. Cells marked ● are first-class capabilities; ○ means the capability is absent; · means partial or out-of-scope. The point of the table is not to argue Zegel is "better" overall — Wazuh is far more capable as a SOC, Veeam far more capable as a backup. It is to show that *the specific column "tamper-evident, witnessed, post-quantum, SMB-priced, Dutch-shaped" has only one cell filled in.*

Capability	Wazuh / AIDE	Sigstore / Rekor	Hyperledger Fabric	Veeam / Borg	Zegel
File integrity monitoring on disk	●	○	○	·	●
Tamper-evident append-only log	○	●	●	○	●
Independent third-party witnesses	○	●	·	○	●
Post-quantum signatures (NIST FIPS 204)	○	○	○	○	●
Offline, vendor-free verification	○	●	○	○	●
Backups / file recovery	○	○	○	●	○
Real-time intrusion detection	●	○	○	○	○
EU-sovereign hosting and jurisdiction	·	·	·	·	●
Dutch-language NIS2 reporting out of box	○	○	○	○	●
Priced for an SMB (≤ €5/endpoint/month)	●	●	○	·	●
Open-source agent	●	●	●	○	●

● *first-class capability* · *partial or out-of-scope* ○ *absent.*

Where the gap actually is

The honest read of the matrix is: every individual capability Zegel offers is offered somewhere already. None of it is novel in isolation. What is missing in the existing market is a product that combines *file integrity monitoring* with *witnessed tamper-evidence*, ships with *Dutch- language NIS2 packaging*, runs on *EU-sovereign infrastructure*, and prices itself for an SMB. The closest cousin (Sigstore) targets a different audience entirely. The closest local alternative (Wazuh) lacks the witnessed log. The original Privachain proposal (Hyperledger Fabric) chose a heavier primitive for the same outcome. Backups are complementary, not competitive.

That is the gap, and that is what we are building Zegel to fill.

7 What only Zegel does — our USP

The unique selling proposition of Zegel is not in any single capability. Each individual primitive — hashing, hybrid signing, witnessed Merkle log — is borrowed from existing open-source work and is, in some form, available elsewhere. The USP is the *specific combination* of three axes, each of which is hard to replicate independently and which together no incumbent has bothered to assemble.

Axis one: the cryptographic substrate

Hybrid post-quantum signatures (Ed25519 + ML-DSA-65) over a witnessed RFC-6962-style transparent log, with offline, vendor-free verification. Each piece of this is open-source commodity. Together it produces audit evidence that survives both compromise of the operator and the eventual arrival of large-scale quantum computers. **This axis has the shortest moat**: any competent team could replicate it in three to six months. We claim no defensibility here, only correctness.

Axis two: jurisdiction and operations

Hosted on Dutch infrastructure (Hetzner FSN, Greenhost), under Dutch jurisdiction, with Dutch-operated witnesses (SURF, NLnet Labs, a notaris), targeting Dutch banking and identity rails (iDEAL, Mollie, KvK) for billing and onboarding. **This axis is moderately defensible**. A US or UK competitor cannot credibly claim EU sovereignty, and no other team is positioned to recruit a Dutch witness federation in the next twelve months — that recruitment depends on individual relationships and Dutch-language credibility, not capital. A German, French, or Belgian competitor could replicate it regionally, but not in the Netherlands without us being already entrenched.

Axis three: NIS2-shaped audience packaging

Dutch-language onboarding wizards, NIS2 Article 21 mapping that an auditor opens and immediately recognises, branded report templates that drop into a compliance binder without modification, integration with the Dutch IT-MSP channel as resellers and integrators, and a price point — €2 per endpoint per month for SMBs — that does not need procurement sign-off. **This axis is the deepest moat**. It is built from years of Dutch-language compliance work, MSP relationships, and an opinionated stance on which exact NIS2 clauses Zegel does and does not address. The cryptography is table stakes; the packaging is the product.

Why incumbents do not fill this gap

The natural question for a grant reviewer or design partner is *why hasn't Wazuh / Tripwire / a Veeam-and-Sigstore bundling done this already?* Three structural reasons.

First, **the audience is too small for an enterprise vendor**. A 100-employee Dutch SMB at €2 per endpoint per month is €2 400 per year. Tenable, Tripwire, and Hyperledger consortia all need orders of magnitude more revenue per logo to justify a sales motion. They will not come down-market.

Second, **the audience is too compliance-shaped for an open-source project**. Wazuh and AIDE are excellent open-source FIM tools, but they are run by hobbyist sysadmins and security teams. They have no Dutch-language NIS2 onboarding because that is not what their users want. The compliance packaging is a paid-product motion that an open-source community will not naturally produce.

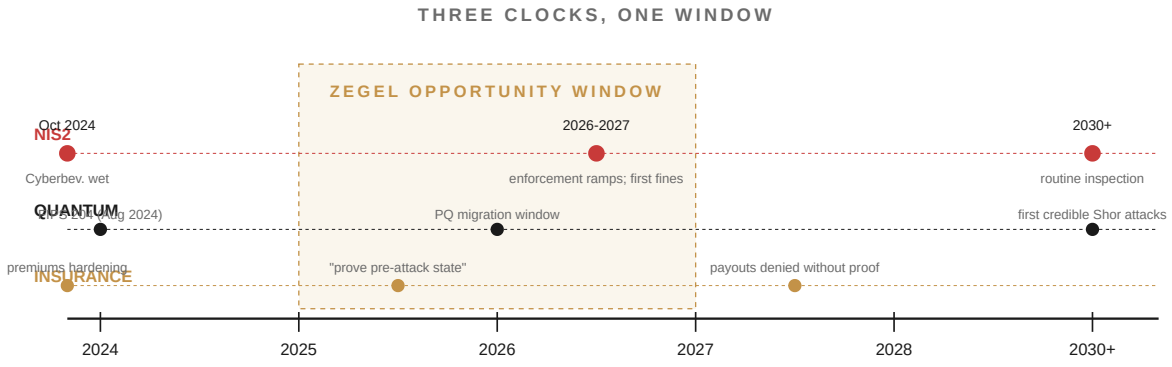
Third, **the audience is too Dutch for a US or UK vendor**. The wedge is not the technology; it is the Dutch SMB who reads a Dutch contract, calls a Dutch accountant, and pays in iDEAL. A US vendor's onboarding will always be in English, billed in USD, and shaped for SOC 2 rather than NIS2. That is a wedge open to a Dutch team and closed to anyone else.

The combination of these three structural barriers is why a tiny Dutch-operated team can credibly take this market against vastly better-resourced competitors, and why the open-source strategy is the moat rather than the threat: by publishing everything, we lower the trust barrier with auditors and witnesses, while the actual operational defensibility lives outside the source code.

> **THE SINGLE SENTENCE USP**
>
> *"Zegel is the only file-integrity product designed, operated, witnessed, and packaged specifically for a Dutch SMB to install in thirty minutes and produce auditor-ready NIS2 evidence the same day, with cryptography that will still hold in 2040."*

8 Why now — the urgency

Three independent clocks are running at the same time. Each one would, on its own, justify building Zegel during the next eighteen months. The fact that they are all running together is what makes the window unusually narrow and the opportunity unusually clear.



All three clocks force the same SMB decision in the same eighteen months.

Diagram 9 — The three urgency clocks: NIS2 enforcement, quantum migration, and cyber-insurance hardening.

Clock one — NIS2 enforcement ramps

The Cyberbeveiligingswet entered into force in October 2024, but Dutch regulators (the *Rijksinspectie Digitale Infrastructuur*, RDI) followed the standard EU pattern: law in year one, written guidance in year two, inspections in year three. We are at the end of year two. The first formal RDI inspections of essential and important entities are expected in 2026. Fines of up to **€10 million or 2% of global turnover** are administrative, not court-ordered, and are issued unilaterally by the regulator. The first round of fines will set the template for the next decade. SMBs that can produce cryptographically grounded NIS2 evidence on day one of an inspection will receive far better treatment than those that cannot.

Clock two — quantum cryptography migration

In August 2024 NIST finalised the first three post-quantum cryptographic standards (FIPS 203, 204, 205). The Dutch *Nationaal Cyber Security Centrum* has issued advisory guidance recommending hybrid signing schemes for long-lifetime data. The European Central Bank, the Dutch financial sector, and any data with a useful lifespan exceeding ten years should already be migrating. The risk is not that quantum computers exist today; it is that adversaries can *store* classical-signed data today and break the signatures retroactively after large-scale quantum hardware arrives — currently estimated at 2030 onward. Audit trails that need to hold up in court ten years from now must already be post-quantum-signed today. Zegel is, by construction.

Clock three — cyber-insurance hardening

Dutch SMB cyber-insurance premiums have risen 50–200% since 2022. Underwriters increasingly require, in their standard policy language, *proof of pre-attack file state* before paying out a ransomware claim. Several recent denials have hinged on the inability of the insured to demonstrate that the restored files were not a "tainted" copy seeded by the attacker. The market is moving toward *discount-or-deny*: if you can demonstrate tamper-evident integrity logging, you get a 10–30% premium discount; if you cannot, your claim faces extra scrutiny. Zegel is the cheapest and quickest way for an SMB to land on the right side of that line.

The eighteen-month window

The opportunity window — when an SMB urgently needs the product, the cryptography is mature, and competitors have not yet noticed — is roughly January 2026 through mid-2027. After that, larger compliance vendors will catch up, the witness federation idea will be cloned, and the Dutch-language packaging advantage will erode. Before then, the SMB market is not yet feeling enough pressure to buy. Zegel needs to have at least one anchor organisation contractually mandating it across its supply chain, and at least ten paying SMB design partners, before that window closes. That is the unit of urgency the architecture roadmap is built around.

9 Vision and roadmap

Zegel's success is measured by a single sentence in 2030: *"a Dutch SMB owner can answer the question 'are these your original files?' in thirty seconds, with a receipt the auditor opens immediately."* The path to that sentence is three concrete stages.

Year one (2026) — credible foundation

- **10 paying design-partner SMBs** across logistics, accountancy, and zorg-toeleveranciers. Each contracted at the €2.5k NIS2-onboarding pakket, with the agent running on their actual file servers.
- **NLnet NGI Zero grant secured** (~€30-50k), funding cryptographer review and the first nine months of full-time engineering.
- **Working ingest in Dutch hosting** (Hetzner FSN or Greenhost), backed by Trillian and Postgres. First witness signing on it, even if only NLnet Labs.
- **Public source repository at `zegel.org`** with the explainer, the demo, the architecture, and the first external code review on record.

Year three (2028) — operational federation

- **200 paying SMBs**, mostly arriving via the anchor-organisation supply-chain motion: a single hospital, gemeente, or large retailer signing a contract that mandates Zegel across thirty to fifty of its suppliers at once.
- **Three independent witnesses operational** — SURF, NLnet Labs, and a notaris koepel — co-signing the log root every hour, monitored publicly.
- **Sovereign Tech Fund maintenance contract** funding ongoing protocol work, security audits, and supply-chain hardening of the agent binary releases (Sigstore, reproducible builds).
- **Dutch NIS2 lawyer's published opinion** confirming Zegel receipts as adequate Article 21(f) effectiveness evidence — a freely-cited reference document for every future buyer.
- **Annual recurring revenue around €500k**, cleanly self-funding the team without further dilution.

Year five (2030) — Dutch standard

- **Over a thousand SMBs** across NL and BE. Zegel is the obvious off-the-shelf answer to NIS2 file-integrity obligations for organisations under 500 employees.
- **Receipts treated as presumptively valid** evidence by AVG, RDI, and Belastingdienst. A Zegel receipt on a file is, in practice, a complete answer to "is this the original?"
- **The witness federation expands beyond NL** to include a Belgian or German notary, opening the cross-border anchor-org motion.

- **Open-source primitive forked by other countries** for their own SMB sectors. Zegel-the-company runs the Dutch instance; the technology becomes a small EU public good.

What the public benefit looks like, concretely

Of the roughly fifty thousand Dutch SMBs hit by ransomware each year, an estimated 70-80% currently cannot produce defensible pre-attack evidence to insurers, regulators, or customers. If Zegel reaches the year-three milestone of 200 paying SMBs and one anchor organisation pulling thirty to fifty suppliers, roughly **500 to 1 000 SMBs annually** would cross the threshold from "we believe so" to "here is the receipt." If the year-five milestone is reached, the figure is in the tens of thousands. The macro- level effect is a measurable reduction in the disputed-payout rate of cyber insurance, faster incident forensics across Dutch supply chains, and — most importantly — a real answer to the central NIS2 question of whether the law's effectiveness-checking obligation can be met practicably by small businesses without a SOC.

10 Try it yourself

Three things you can do in the next half-hour to verify, with your own eyes, that everything described in this document is real, working, and reproducible.

1. Watch the demo run automatically on every commit

Every push to the Zegel repository triggers a GitHub Actions workflow that, on a fresh Ubuntu virtual machine, builds the Rust agent and verify CLI, builds the Go ingest, runs the cryptographic unit tests, then runs the end-to-end demo script — sealing three sample files, verifying all three receipts offline, and finally tampering with one file to show the verifier reject it.

A green check next to a recent commit on `github.com/HWqs/privachain/actions` is your cryptographic proof, performed by GitHub's machines, that the entire pipeline still works. A red X means we shipped a regression. There is no installation required to read this signal.

2. Open the animated visualization in any browser

`web/index.html` in the repository is a single self-contained HTML page that animates the sealing flow: files appear on disk, get hashed, get signed twice, fly into a growing Merkle tree, and produce a receipt. Every fifth file in the loop is a tamper-detection moment: an existing file changes on disk and the verifier rejects it with a clearly-marked `[FAIL]`.

This page is intended for the eventual hero section of `zegel.org` and works as a stand-alone explainer. It needs no server: open the file directly in a browser.

3. Run the demo locally

On Linux, macOS, or WSL the demo script builds and runs everything from a fresh checkout:

```
git clone https://github.com/HWqs/privachain
cd privachain
./scripts/demo.sh
```

You will see the agent seal three files, the verifier accept all three, and the verifier reject a deliberately tampered file. The whole demo runs in under thirty seconds on a recent laptop. Requires Rust, Go, and a C compiler installed; the script tells you if anything is missing.

Further reading in the same repository

For grant reviewers, design partners, lawyers, and anyone wanting to dig deeper, the rest of the repository contains:

- **`README.md`** — entry point and repository tour.
- **`concept.md`** — the original Dutch deck, with an English senior-engineer review of which parts to keep, cut, or rewrite.
- **`architecture.md`** — the full v0 architecture: components, threat model, data shapes, roadmap, NIS2 mapping.
- **`use-cases.md`** — past real-world incidents (Diginotar, Maastricht University, SolarWinds, Maersk, Colonial Pipeline) where Zegel-style proof would have changed the outcome.
- **`funding.md`** — solo-developer- accessible grant landscape, ranked by realism.
- **`common/`, `agent/`, `verify/`, `ingest/`** — all of the actual source code referenced throughout this document.

How to reach us

Project repository: `github.com/HWqs/privachain` (will become `github.com/HWqs/zegel` at public launch). Issues, pull requests, and design-partner enquiries welcome.

For grant reviewers: the architecture document and this explainer together cover the technical, market, and timeline content of an NLnet NGI Zero or Sovereign Tech Fund application. Cite either as the complete artefact.

For SMBs interested in becoming design partners: the closed- beta is free for the first ten organisations and includes the onboarding pakket at no charge. Email the maintainer through the GitHub repository.

> **IF YOU READ ONLY ONE PARAGRAPH OF THIS DOCUMENT**
>
> Zegel produces tamper-evident, post-quantum-signed, witnessed proof that a file existed unaltered at a moment in time, verifiable offline by anyone with the file and the receipt. It is built specifically for Dutch SMBs facing NIS2 obligations, hosted under Dutch jurisdiction, witnessed by Dutch organisations. The cryptography is real, the demo runs in CI on every commit, and the eighteen-month opportunity window is open now.