An Anchor Outside the Industry

Allan Watkins

Laws of Robots series

Version 1.2 — 2 November 2026

Companion to "The Maker Has Always Graded Its Own Homework," version 1.2 — 31 October 2026, §§1, 6–7 and Appendix, including reference [30].

Abstract

The companion paper compares five historical episodes across four industries and argues for independent scrutiny, access to relevant evidence, and authority to require correction. It does not establish a universal failure of self-assessment or one guaranteed remedy. Autonomous machines already have a history of harm. This paper develops one proposed response as five anchor pillars, then grades an existing certification anchor against them in public — reporting which pillars run today as checkable fact (outsider verification over an offline floor; a bounded claim whose disclaimer travels in the wire format; structural separation with one seam disclosed) and which remain the author's intent (institutional independence, stated flatly as unbuilt). The author operates both the certified system and the certifier; the paper is built around that conflict, dividing every claim into checkable and vouched, and it closes with a numbered exercise the reader can run with the certifier's servers unreachable. Its purpose is to strengthen checkable assurance before further harm, while leaving the institutional form open.

§1 — The institutional task

The companion paper's summary table ends with autonomous machines and an ongoing task: independent access to evidence and effective correction. It acknowledges the fatal 2018 Tempe automated-driving collision and existing outside scrutiny. Five historical episodes span four industries: corporate accounting supplies both the accounts and audit-profession rows. Their different failures and reforms inform the question here: how can a certification anchor support scrutiny, challenge and correction before further harm?

This paper examines one proposed response, treating independent scrutiny and checkable evidence as design requirements rather than a universal historical fix. It evaluates a certification anchor that exists today: partly as running, publicly checkable machinery; partly as versioned commitments; partly as intent that remains unbuilt and says so.

The companion distinguishes official findings from the author's institutional argument. This paper applies the same discipline: it points to what readers can check and labels what rests on the author's account.

§2 — The author's conflict, stated first

The author of this paper is the single operator of both the certified system and the body that certifies it. That is a conflict the previous paper argues must be addressed through independent scrutiny, and no formulation of it removes the conflict.

So the conflict is the method. Because the author's word carries so little weight here, this paper divides every claim into two kinds and never blends them. A **checkable claim** can be tested by the reader today against a public artifact — a live endpoint, a signed file, a fetchable document — with no trust in the author required. A **vouched claim** rests on the author's account of things the reader cannot see. Checkable claims appear under the label OPERATING. Vouched claims appear under DESIGN-INTENT, or, in the rare case where a vouched sentence earns a place inside an operating section, it is flagged in-line as exactly what it is.

The anchor's own founding document works the same way, and it is this paper's first exhibit. The certifier's public charter carries a section titled "What is deferred, stated honestly," listing plainly what the body does not yet have — full institutional independence among the items, in those words: today a single operator oversees both the certified system and the certifier. A trust body that implied completeness where none exists would repeat the self-certification error this series documents; naming the gaps is not modesty, it is the entry fee.

One more sentence belongs here, flat, before anything running is described: nothing in the period this paper documents advanced actual independence. The machinery that came online — described in §4 — made the anchor more checkable. It did not make it more independent.

§3 — The anchor model as a specification

The previous paper argues for independent examination of evidence, judgment sufficiently independent of interested parties, and authority to require correction. Its reforms took different forms, and it prescribes no single institution or guaranteed remedy. The anchor model here proposes **independence of the maker**, **verification by outsiders**, and **structural rather than promissory safeguards**. These are requirements informed by the historical comparison, not a universal historical formula.

The anchor's charter adds a requirement: the certifier must be **incapable of disabling** what it certifies. An inspector with a kill switch would be a second master. This is the model's design requirement, not a conclusion established by the historical cases alone. The charter's full commitment set is independent, owner-defined, verifiable, and never safety-disabling.

That is the specification. The remainder of this paper grades one existing anchor against it, in public, including the failing grades.

A definition, used from here forward: in this paper, a “pillar” is a discrete, named structural requirement of the independent anchor model. Each pillar can be evaluated, verified, or implemented independently while contributing to the overall structure.

§4 — The anchor, pillar by pillar

Each pillar below opens with one of two labels. **OPERATING** means the sentence can be tested today against a public artifact — an endpoint, a signed file, a fetchable document — without taking the author’s word for anything. **DESIGN-INTENT** means the requirement is specified but not built; what follows that label is the author vouching, and it is written so it cannot be mistaken for anything more. No paragraph mixes the two.

4.1 Verifiable by outsiders — OPERATING

Of the five pillars, this one is the keystone. If outside verification fails, nothing else in the model can be checked, and the structure reverts to the pattern the previous paper documented: the judged party grading its own homework.

Any third party can confirm a robot’s certification status at a public endpoint on the certifier’s domain. The route is unauthenticated by design; no account, no key, no relationship with the certifier is required. The response is a bounded result: the certification status, a genuineness confirmation against the certifier’s published key, a chain-integrity confirmation, and — verbatim, in every response — a statement of what the result does not assert. The cornerstone element carrying that last field is the subject of §4.2.

The endpoint, however, is deliberately not the guarantee. It is a courier — convenience over an offline floor, never a gate. The inputs to a verification are themselves fetchable: the signed verification bundle, the published trust-anchor set, and a signed, timestamped revocation statement. A relying party can download those files, re-host them on infrastructure it controls, disconnect from the network entirely, and reach the same bounded result with an offline verifier. For the same signed inputs, trust anchor, and verification time, the hosted and offline checks return the same shared bounded values; their output formats differ. That equivalence — the certifier’s server switched off and the answer unchanged — is the demonstration that the hosted path adds convenience and no authority. Appendix A turns it into a numbered exercise the reader can run.

Revocation closes the one dependency a purely offline check would leave open. Whether a signing key has since been pulled is inherently a current-state question, so a verifier holding stale materials could accept a clearance signed under a revoked key. The mechanism: revocation travels as a signed, timestamped statement, fetchable from any mirror rather than from the certifier’s live server specifically. Withholding a fresh statement cannot forge a valid status — the last signed statement stands, and its timestamp makes staleness visible to the relying party. The certifier’s silence degrades freshness; it never manufactures trust.

One further commitment belongs in this pillar because it is falsifiable. The published Independent Verification Specification claims to be sufficient, on its own, for a third party to build a verifier from scratch — no reference implementation required, no private clarification available. Anyone may attempt the exercise; the claim stands or falls on whether they succeed.

A checkable promise is the strongest sentence a conflicted author can offer — which is why this paper leans on it rather than on assurance.

4.2 A bounded claim — OPERATING (cornerstone)

A certification here asserts two narrow things and refuses everything else. First, reasoning integrity: the machine's reasoning engine has not suffered an integrity fault that would cause it to de-certify itself. Second, genuineness: the clearance was issued by the certifier — whose signature the robot cannot forge, because the robot holds only the public anchor. "Verified genuine" means *the certifier issued this clearance*, not *the robot vouched for itself*.

Deliberately outside the claim: whether the recorded decisions were good ones, whether the owner's rules are wise, whether the machine is safe in any general sense. The certification attests a property, not a product — and integrity of the record is not wisdom of the conduct. A narrow attestation is also a durable one: it is not invalidated by every change in rules, curriculum, or configuration, and it cannot be inflated by a sales page, because of where the boundary is written.

It is written into the wire format. Every result — public or authorized, hosted or offline — carries a `DoesNotAssert` field: the certifier's statement of what verification does not claim, verbatim, machine-delivered alongside the status it qualifies. A consumer that surfaces a result to a human is required to surface the negative with it. The disclaimer travels in the protocol, not in the marketing — the architecture is arranged so "verified" cannot quietly be presented as "endorsed."

One further property is asserted here without a public check — the served negative is compiled from a pinned constant, not read from editable configuration, so the boundary text cannot drift through an operations change; the reader has only the author's word for this, and the sentence is labeled accordingly.

The text of that boundary is not this paper's paraphrase. It exists as a published, versioned statement of record — the Bounded-Claim Statement — which fixes, in the certifier's own words, what a stamp asserts and what it deliberately does not.

4.3 Structural separation — OPERATING, with a disclosed seam (linchpin)

The public verification surface lives in a structurally separate tenant; it holds only the public anchor; it has no vault, no path to the certifier signing key, and no path to the certified party's audit data. Separation of duties here is not an org-chart line — it is a boundary the infrastructure

enforces, which is why this element is the linchpin of the model's near-term credibility: it is what makes "fixer ≠ certifier" physical rather than rhetorical while the institutional requirement (§4.4) remains unbuilt.

Part of this is directly checkable. The verification domain, routes, payloads, and published anchor carry no operator or company name anywhere; the root of trust is a key identifier and an algorithm tag, nothing more. A reader can inspect every public identifier and confirm that no durable name binds the certifying role to the party being certified — a deliberate design rule, so that a future operator, or a future rename, strands nothing.

Part of it is topology the reader cannot see from outside, and this pillar owes the reader precision about which is which. The tenant boundary and the surface's contents are asserted from the build record, not independently observable. And one seam remains open: custody of the certifier's signing key has not yet crossed the tenant boundary. The key sits sealed in a vault inside the same tenant as the certified system's infrastructure — reachable by no path from the public surface, but not yet housed under the certifier's own roof. Its migration is a named, deferred step, gated on a key-custody ceremony, and it may never prove necessary; moving a live signing key is itself a risk. Disclosing that seam here, rather than rounding the claim up, is §2's method executed: the separation that exists is stated exactly, and the separation that does not is named.

4.4 Independent of the maker — DESIGN-INTENT

No quorum exists. Signing authority rests with one person. There is no foundation, no board, no separate legal entity, and no named operator behind the abstract certifying role. The multi-party signing scheme, the foundation governance modeled on ICANN's, and the legal separation of certifier from maker are specified — and unbuilt. The separation that operates today is architectural and tenancy-level, not institutional. Nothing in the period this paper documents advanced this requirement. When it is built, it will be checkable the way the pillars above are; until then, no sentence in this paper should be read as a claim that independence has been achieved.

4.5 Never safety-disabling — OPERATING

Verification reports; it never controls. The published interface defines no channel by which a verification outcome seizes, disables, or commands a machine: the wire formats carry a status, its supporting confirmations, and the explicit negative — no actuation, no remote authority. A revoked or failed status changes what the record says, visibly, to anyone who checks; what happens next flows through people — owners, insurers, counterparties — not through a kill switch held by the certifying body. An anchor that could disable the machines it grades would be a second master, not an outside referee, and the model refuses the role by construction.

§5 — The line between public and private

A reasonable reader will ask why the checking stops where it does. Anyone can confirm the stamp; no one but the owner, and parties the owner authorizes, can read the record behind it. This section states where that line sits and why it is a published boundary rather than a defensive one.

The access rule is itself part of the versioned governance record, in the certifier's words: confirming the public certification claim does not grant the ability to read the robot's record. A stamp, not the contents. The engineering enforces the same line the commitment draws. The public tier answers with a bounded result and a single integrity confirmation — it carries no enumerable timeline of events, so an outside observer learns that the record is intact, not what is in it or when anything happened. The full record is a separate, authorized tier, for the owner and the auditors the owner admits. And even on that tier, the chain carries ciphertext only: verification never requires decryption, so checking the record's integrity and reading its contents are different acts with different keys.

Two small details reward the forensic reader. First, when no verification materials exist for a queried identifier, the public routes fail with a plain not-found — never a demand for credentials — which is a quiet, checkable proof that the open tier is genuinely open rather than a login wall with a brochure in front of it. Second, the boundary text and the access rule come from the same versioned commitments as everything else in §4: the line between public and private is not this paper arguing for privacy, it is the paper quoting a rule that was published before the paper was written.

That is the whole defense, and it is deliberately short. The public surface is fair game precisely because it is public — mirror it, probe it, rebuild the verifier from the specification and try to catch the hosted answer lying. The certified system's internals are not on the table, for the same reason a homeowner's audit trail is not: the boundary protects the owner of the record, not the reputation of the certifier.

§6 — What this does not prove

The operating sections above are written in a confident register because their claims are checkable. That register must not be allowed to imply more than it says, so the limits go here, in one place, unqualified.

One operator's own machine population is certified. No third party has adopted this anchor; no maker submits to it; no insurer relies on it. Nothing in this paper is evidence of market traction, and the operating label means "running and checkable," never "in use by others."

A valid certification does not mean the robot behaved well, will behave well, or is safe. It means the record's integrity holds and the clearance genuinely came from the certifier. Integrity of the record is not wisdom of the conduct; the explicit negative exists precisely because those two are easy to conflate and profitable to conflate.

And the deepest requirement is unmet. However checkable the machinery, the anchor's judgments today trace to one person's key. Every pillar marked OPERATING makes the anchor more inspectable; none of them makes it independent. That grade stays failing until §4.4 is built, and this paper has no interest in softening it.

§7 — Close: an anchor that holds when the author is gone

There is one design property that unifies everything marked OPERATING in this paper: none of it requires the author. The endpoint can go dark and the offline verifier still answers. The certifier can fall silent and the last signed statement still stands, its age visible. The specification is claimed sufficient for a stranger to rebuild the whole check from paper. The author's honesty was made non-load-bearing on purpose, by someone who had read enough history to know that honesty is the first thing an institution loses and the last thing it admits losing.

The companion argues for independent examination, access to evidence, and authority to require correction. It leaves the institutional form open and acknowledges prior harm. This paper adds one anchor's outsider-verifiable machinery, available for examination while institutional independence remains unbuilt. Neither the record nor the model establishes a unique remedy or guarantees safety. Appendix A lets the reader test the bounded operating claims.

Appendix A — Verify this paper's claims yourself

Rehearsed against GRRC's anonymous public service on 31 October 2026. The service belongs to GRRC; the LOR website is not the certification verifier.

The example below uses an already-public registry identifier, `d1479628-4b6a-458b-bb57-ac4c5cae4a36`. No account, owner record, private key, or private Audit history is needed. The human-facing lookup is verify.grrc.ai. The API base is `https://verify.grrc.ai/api/v1/grrc/verify`; the identifier and optional resource suffix follow it.

Before starting. These commands use PowerShell 7.5 or later, `curl.exe`, and the GRRC reference command-line verifier, `Res.Tools.GrrcVerify`, with the .NET 10 runtime. Obtain the verifier's complete build output before going offline and put it in a local folder named `verifier`. At this revision, GRRC's How to Verify page publishes the independent specification but still describes the packaged verifier download as coming. The reference tool has been tested locally; this appendix does not claim that its public download is already available. A reader without the reference build must implement the published specification or obtain the package when distributed.

1 Download the public result and inputs

Run these commands in an empty working folder while connected. Stop if a download fails.

```
$hostUrl = 'https://verify.grrc.ai'
$robot = 'd1479628-4b6a-458b-bb57-ac4c5cae4a36'
$base = "$hostUrl/api/v1/grrc/verify/$robot"
curl.exe --fail --location "$base" --output public-result.json
curl.exe --fail --location "$base/bundle" --output bundle.json
curl.exe --fail --location "$base/anchors" --output anchors.json
curl.exe --fail --location "$base/revocation" --output revocation.json
```

Read the saved result and retain its verification time as text. `-DateKind String` preserves the timestamp's offset and fractional seconds.

```
$public = Get-Content -Raw public-result.json | ConvertFrom-Json -DateKind String
$asOf = $public.asOfUtc
$public.doesNotAssert
```

Read the entire `doesNotAssert` statement before continuing. A valid result is a bounded certification claim, not an endorsement of the robot's conduct or the owner's rules.

2 Prepare the public anchor and inspect the evidence date

The anchors endpoint returns JSON containing a base64 DER SubjectPublicKeyInfo key. The command-line verifier accepts that public key in PEM form; no private key is involved. For this tested example, select key ID `grrc-certifier-root-v1` and convert it as follows.

```
$keyId = 'grrc-certifier-root-v1'
$anchorSet = Get-Content -Raw anchors.json | ConvertFrom-Json
$anchor = $anchorSet.Anchors | Where-Object KeyId -EQ $keyId
if (@($anchor).Count -ne 1) { throw 'Expected public anchor not found uniquely.' }
'-----BEGIN PUBLIC KEY-----' | Set-Content -Encoding ascii anchor.pem
$anchor.PublicKeySpkiBase64 | Add-Content -Encoding ascii anchor.pem
'-----END PUBLIC KEY-----' | Add-Content -Encoding ascii anchor.pem
```

The downloaded key matched the public PEM in the reference verification package during this rehearsal. Establish trust in the published anchor independently before relying on it: downloading a key and a signed claim together proves consistency under that key, not the key's identity by itself.

Confirm that the separately downloaded revocation statement is the one inside the bundle, then decode its signed content to inspect its timestamp. Decoding is inspection; the verifier performs the signature check in the next step.

```
$bundle = Get-Content -Raw bundle.json | ConvertFrom-Json
$revocation = Get-Content -Raw revocation.json | ConvertFrom-Json
if ($bundle.AnchorStatementPayloadBase64 -cne $revocation.AnchorStatementPayloadBase64) {
    throw 'Signed statement changed between downloads; fetch a consistent set.'
}
function Decode-B64Json($text) {
    $bytes = [Convert]::FromBase64String($text)
    [Text.Encoding]::UTF8.GetString($bytes) | ConvertFrom-Json -DateKind String
}
$envelope = Decode-B64Json $revocation.AnchorStatementPayloadBase64
$statement = Decode-B64Json $envelope.signedContent
$statement.asOfUtc
```

In the tested inputs, both the checkpoint and the anchor/revocation statement were signed as of `2026-06-21T04:18:46.3114194+00:00`. The endpoint's later `asOfUtc` is the time of verification, not a newly signed statement of currency. A saved statement cannot establish that no later revocation or certification event exists. Its visible age matters to the relying party.

3 Run the verifier using only local files

Disconnect the laptop's network for a physical offline demonstration. The verifier itself reads local files and makes no network calls. Keep the same PowerShell session, or reload `$keyId` and `$asOf` from the saved files as above, then run:

```
dotnet .\verifier\Res.Tools.GrrcVerify.dll `
    --artifact .\bundle.json `
    --anchor-pem .\anchor.pem `
    --anchor-key-id $keyId `
    --as-of $asOf
$LASTEXITCODE
```

Exit code `0` means `Valid`; `3` means the check ran and returned a bounded non-valid status; `1` means indeterminate or a usage error. Read the status and diagnostics, not the exit code alone. Fixing `--as-of` reproduces the saved online check; it is not a claim about the current time. To check the grant's lifetime at the time of a later run, omit `--as-of` and read the resulting verification time.

4 Compare the common fields and try a rejection

Compare `robotUuid`, `certStatus`, `overallVerified`, `genuinenessConfirmed`, `genuinenessKeyId`, `chainIntegrityConfirmed`, and the full `doesNotAssert` text with the saved JSON. The CLI labels chain integrity as `Chain intact` and prints enum names and booleans with different capitalization. Compare those values, allowing for presentation and line endings; the CLI transcript and API JSON are not byte-identical documents. Additional CLI fields, such as sequence and revocation currency, have no corresponding field in this public JSON result.

The rehearsal returned `Valid` with exit code `0`; overall verification, genuineness, and chain intactness were true, and the key ID was `grrc-certifier-root-v1`. The common fields matched, including the complete limitations text. The CLI also reported sequence `664` and revocation currency true under the supplied signed statement. Changing a grant-signature byte caused `GenuinenessFailed` with exit code `3`. Using `2027-06-18T00:00:00+00:00`, after the grant's expiry, caused `Expired` with exit code `3`.

These checks used the unchanged reference CLI and saved local inputs in a restricted execution environment: a control HTTP request to GRRC failed, while local verification returned `Valid`. The workstation's physical network adapter remained connected; readers can repeat the physical-disconnection step above. The downloaded public bundle contains an empty chain segment and no robot audit public key. Its integrity result therefore verifies the certifier-signed checkpoint; it does not independently walk the private Audit chain.

5 Reconnect and look for newer evidence

Re-fetch the result and all three inputs into separate files. A new verification timestamp is normal. If the signed inputs or bounded status changed, inspect the new statement dates and repeat the local check on that new set. Do not require a later response to be byte-identical to the earlier one, or treat a matching old response as proof that nothing happened while disconnected.

Appendix B — Pillar status at a glance

Pillar	Status	Evidence artifact	What a reader can check	What remains
Verifiable by outsiders	OPERATING	Public endpoint · verification bundle · anchor set · revocation statement · Independent Verification Specification	Fetch, mirror, verify offline; matching bounded values (Appendix A)	—
Bounded claim	OPERATING	Bounded-Claim Statement · `DoesNotAssert` in every wire result	Read the negative in the live response; compare against the published statement	—
Structural separation	OPERATING, seam disclosed	Public surface identifiers · build record (asserted)	No operator name in any durable identifier	Certifier key custody across the tenant boundary (deferred, ceremony-gated)
Independent of the maker	DESIGN-INTENT	Charter §I (honest deferral)	That the gap is stated, in the certifier's own charter	Quorum · foundation · legal entity · named operator — all of it
Never safety-disabling	OPERATING	Published interface (report-only wire formats)	No control channel exists in any public format	—

Allan Watkins

Laws of Robots series

Version 1.2 — 2 November 2026

This document is part of the Laws of Robots governance whitepaper series. The version of record is maintained under version control. All commitments are owner-authored and subject to the published change discipline.