KNOBE Protocol v1: A Context Protocol for Responsible Knowledge Movement

Carrying source, history, limits, and obligations across human and AI systems

David Kyle · University of California, Davis · knobe.org

Public release, June 28, 2026

Abstract

KNOBE Protocol v1 is an open protocol for responsible knowledge movement. It defines a plain-text `.knobe.md` file that lets a knowledge object carry its readable content together with a machine-legible, hash-sealed record of attribution, source relations, transformation history, fidelity limits, use conditions, accessibility adaptations, quarantine status, and integrity checks.

The problem KNOBE addresses is simple. When knowledge moves between people, platforms, course systems, archives, AI tools, summaries, and reports, the words often survive while the conditions that made them responsible to use do not.

KNOBE does not prove truth, authorship, authorization, or trustworthiness. Its hash proves only that the sealed payload has not changed since sealing. Attribution is declared, not independently verified, and new or external KNOBEs should be treated as quarantined until a human or governed system marks them trusted.

KNOBE is informing infrastructure, not controlling infrastructure. It reports what traveled with the object so that humans, institutions, and AI systems can decide responsibly. KNOBE's practical claim is that knowledge objects need a portable way to carry their source, history, limits, and obligations across human and AI systems. Its deeper theoretical claim is that doing so preserves objecthood under compression.

1. The Problem: Fragments in Transit

KNOBE begins from a practical failure in knowledge work: documents move, but the conditions that make them responsible to use often do not. A file gets copied, pasted, summarized, exported, adapted, or fed into an AI system. The visible words keep moving, while attribution, source relations, consent limits,

transformation history, fidelity limits, and use conditions fall away.

This is not a failure of bad faith. It is a structural failure of handoff. Knowledge has always depended on compression: a citation compresses a lineage, a credential a training history, a reputation a body of work, an abstract an article, a syllabus line a course. Compression is what makes knowledge portable at all.

But compression creates a recurring danger: the artifact survives while the conditions that made it interpretable disappear. KNOBE addresses that danger directly, by giving a knowledge object a way to carry enough of its own interpretive record with it, inside the artifact itself.

This white paper has one purpose: to explain why KNOBE exists, what problem it solves, and how it can be adopted without depending on any particular platform.

Institutions have always managed compressed knowledge (citations, credentials, rankings, files, reports), deciding which compressions count and who may interpret them. They are, in effect, systems of *authorized decompression*. Their recurring failure is that a compressed object becomes *operationally sufficient*: treated as adequate to act on while the partial, situated field that made it interpretable drops from view. A file becomes a person. A credential becomes competence. A ranking becomes quality. A summary becomes the source. Institutions arose in part to govern that danger, but knowledge objects now move faster than institutional context can travel with them. KNOBE addresses that gap at the artifact layer.

This pattern long predates computing. What has changed is speed, scale, and the role of AI systems in receiving, transforming, and transmitting compressed knowledge objects. Books, articles, notes, prompts, transcripts, clinical encounters, student work, legal excerpts, research records, organizational decisions, and creative works now move through pipelines in which human inspection is intermittent.

What travels through those pipelines is rarely the full object. It is the chunk that fits a context window, the excerpt that pastes cleanly, the screenshot, the table, the summary of a summary. Pipelines select for what is small, portable, and machine-tractable. The context that made the fragment meaningful often does not travel with it, because nothing in the pipeline is responsible for carrying it.

Consider a simple case. A graduate student interviews an elder for an oral-history project. The transcript carries consent terms, a request that certain passages remain unquoted, and the interviewer's contextual notes. A semester later, an excerpt is pasted into an AI assistant to draft a conference abstract. The abstract is summarized for a department newsletter. A sentence from the newsletter is quoted in a grant report.

Every step may be reasonable. No one need have acted in bad faith. But by the fourth step, the words still circulate while the consent terms, the do-not-quote request, and the contextual notes have disappeared. No one downstream knows they ever existed.

This is **context survivorship bias**: the fragment that survives transit is mistaken for the full knowledge object, while the missing context that made it interpretable disappears from view. The name is deliberate kin to survivorship bias in statistics: the analytical error, famously formalized in Abraham Wald's wartime work on aircraft survivability, of reasoning only from the cases that returned and treating the missing ones as if they never existed (Mangel & Samaniego, 1984). In knowledge circulation we interpret the fragments that arrived, and the absence of the rest is invisible by construction.

The obvious diagnosis is wrong. The problem is not compression. Compression is unavoidable, and often the entire point. The problem is **compression without portable interpretive obligation**.

A knowledge object is more than text. It is text plus the relations that make the text interpretable: who produced it, under what constraints, derived from what sources, transformed how and by whom, reliable for which purposes, governed by which fidelity limits, and subject to which conditions of use.

When those relations fail to travel, the result is an **orphaned fragment**: content that still looks like an object but can no longer answer for itself. The excerpt forgets its source. The summary forgets its fidelity limits. The pasted passage forgets whose judgment it embodies.

Institutions experience this as rising coordination cost. Every handoff between a person, a tool, a department, a platform, and an agent requires someone to reconstruct what did not arrive, or else to proceed without reconstructing it. The costs surface as misattribution, unrepeatable analyses, untraceable adaptations, credit drifting away from contributors, and decisions made on fragments mistaken for wholes.

James Gleick's account of the talking drum in *The Information* makes visible a principle central to KNOBE. In that account, drummers signaling across miles of forest did not shorten their messages to push them farther. They lengthened them, wrapping a compressed core in redundant, context-supplying phrases so that meaning could survive the distance.

The point is not unique to drums. Long-distance communication has often depended on meaningful redundancy: sailors using repeated signal flags, telegraph operators relying on routing conventions and confirmation codes, oral traditions preserving stories through formulaic phrases, scientists attaching methods and citations to findings, archivists preserving provenance alongside records. Across these cases, the extra material is not merely extra. It helps the receiver know what kind of message has arrived, where it came from, how to read it, and what not to infer from it.

A single well-chosen word can be a complete and faithful knowledge object. The lesson is that as a signal is compressed and travels farther through more hands, platforms, and AI systems, it needs interpretive support to remain intelligible when it arrives.

KNOBE supplies that support without imposing a rigid schema. Source, transformation history, fidelity limits, use conditions, accessibility lineage, and quarantine posture are common forms of context-rich redundancy, but they are not the only ones. A teacher might include assignment purpose, permitted AI use, or the learning standard a student was trying to meet. A researcher might include consent limits, field conditions, uncertainty, excluded evidence, or why a source was trusted. An accessibility worker might include what was simplified, captioned, translated, preserved, or necessarily lost. A team might include decision history, version rationale, local terminology, handoff instructions, or warnings for an unknown future reader.

The point is not to fill every field. It is to carry enough interpretive support for the object's likely journey, including destinations the author cannot fully predict. KNOBE treats context as a creative responsibility: the maker decides what another person, tool, institution, or AI system may need in order to use the object responsibly without mistaking a compressed fragment for a complete knowledge object.

KNOBE addresses this problem at the only layer that travels everywhere: the artifact itself. Its aim is not to stop compression but to make it harder for a compressed knowledge object to become consequential while pretending it is whole.

---

2. What KNOBE Is

KNOBE is an open protocol for knowledge objects that need to survive handoffs.

A KNOBE is not a platform. It is not a database. It is not a learning management system, repository, rights-management layer, blockchain, DRM scheme, or AI model. It is a plain-text artifact format for carrying context and obligation with a knowledge object.

A KNOBE is implemented as a `.knobe.md` file: one file containing a readable markdown body and a machine-legible, hash-sealed payload. A human can read the document in any text editor. A machine can decode the structured payload. A verifier can check the payload hash. No conversion, no sidecar file, no platform dependency.

KNOBE is a **system of context, not a system of record**. Systems of record (the course system, the ethics board, the repository, the archive, the HR system) each hold an authoritative slice of institutional truth, and KNOBE replaces none of them (Section 7 returns to this). KNOBE carries context *between* those systems: who made the object, what it derives from, what changed, what constraints apply, what fidelity limits govern it, and what humans or agents need in order to interpret it responsibly. Its value is lower coordination cost across people, tools, institutions, and workflows, especially as knowledge increasingly moves through mixed human-machine processes.

Two boundary clarifications matter.

First, KNOBE is not only for AI-heavy workflows. A student who wrote entirely without AI uses the same attribution and process fields as one who used four tools. A researcher working only from archival materials uses the same structure as one using AI-assisted synthesis. The format does not prejudge the workflow. It records what actually happened.

Second, KNOBE is not merely metadata. Metadata describes an object from outside. KNOBE makes selected interpretive obligations travel *inside* the object. Once an object reaches the KNOBE surface, its attribution, source anchors, constraints, intentionality, fidelity limits, and use conditions are not merely *about* the object; they become part of what the object *is* for future use.

Objecthood: known + knowing

Objecthood is not an invented technical label. It names a condition: the one under which something can be recognized and used as a meaningful object rather than a mere thing. In this paper, it means the condition that lets a knowledge artifact remain more than transferable content. A knowledge object is not merely text, data, or media that can be copied. It is an artifact whose internal relations (its sources, making, transformations, limits, obligations, and conditions of use) remain available enough for it to be interpreted, questioned, credited, adapted, verified, and built upon. It is not the same as physical thinghood, and a fragment can move without it: a stripped excerpt is still a thing, but no longer an object that can answer for itself. Objecthood is what lets one act of knowing become a stable input for another.

The issue is older than AI. Writing already changed knowledge by turning thought into durable artifacts that could be revisited, compared, annotated, taught, criticized, and recombined. Once thought becomes an object, it can become input for higher-order thought: thinking about thinking, method about method, commentary about text, systems built from prior artifacts. KNOBE begins from that same premise under contemporary conditions. If knowledge artifacts are now copied, compressed, summarized, adapted, and passed through machines at scale, they need a way to preserve enough of their objecthood to remain responsible building blocks.

A knowledge artifact is not only the *known*: the text, image, file, or output. It also carries the *knowing* that produced it: the labor, position, source relations, constraints, intention, method, and process behind it. Neither alone is enough: the known without the knowing becomes harder to interrogate, extend, credit, or trust, and the knowing without the known leaves no durable artifact to carry forward. Knowledge that travels needs both held together: *known + knowing = knowledge*. What is ordinarily called "context" is just the knowing after it has been cut away and demoted to optional background; a file refuses nothing, so the cut is usually silent.

KNOBE cannot preserve all of the knowing. That would be impossible, and it would defeat the compression that makes knowledge useful. It preserves the parts of the knowing the maker judges necessary for responsible reuse: attribution, source relations, transformation history, fidelity limits, accessibility lineage, use conditions, and quarantine posture. In doing so it holds that cut up to view,

turning it from a silent default into a visible act, and it makes process literacy part of the object itself. KNOBE is pedagogical in this sense: it does not merely store process information for machines but teaches makers and receivers to ask what kind of object they are handling: where it came from, what transformed it, what it preserves, what it loses, who contributed, what obligations travel with it, and what a responsible next use would require.

Accessibility adaptation is the paradigm case. When work is captioned, simplified, translated, described, or otherwise transformed so someone else can use it, the adaptation becomes knowledge work in its own right. It carries judgment, labor, and fidelity limits, the knowing that should not vanish when the object travels on. KNOBE makes that transformation visible and keeps the adaptation tied to the source it was made from.

A note on the name. KNOBE was coined in July 2025 as an acronym, *Knowledge Native Object for Bots and Engines*. The "bots and engines" framing has since proved too narrow: KNOBE is for human handoffs as much as machine ones, and the letters are now best read simply as the protocol's name. Read KNOBE as it functions: an open protocol for knowledge objects that keep their context when they move.

---

3. One Artifact, Three Layers

A `.knobe.md` file is one plain-text artifact with three layers, all simultaneously present:

```
---
{YAML frontmatter}
---
{Markdown body, free-form, no schema}
```

```
-----BEGIN KNOBE B64-----
{Base64-encoded UTF-8 JSON payload}
-----END KNOBE B64-----
```

The **frontmatter** is lightweight YAML readable without tooling: title, author, spec version, license, date, content type, and related fields. It mirrors key payload fields so that a person, or a file browser, can scan the object in seconds.

The **body** is standard markdown: the human-readable content, unconstrained by schema. It can be an essay, field note, transcript, lesson, recipe, protocol, reflection, accessibility adaptation, index, or other knowledge object. This layer is why a KNOBE never requires special software to be *read*.

The **payload block** is Base64-encoded UTF-8 JSON carrying the structured record: attribution, key concepts, version history, privacy level, quarantine status, parents, transformation history, fidelity limits, use conditions, accessibility fields, optional build recipes, and the integrity hashes. Base64 is deliberate (Josefsson, 2006): raw JSON in the body would wreck human readability; Base64 is opaque to casual reading but trivially decodable by any parser. Exact payload markers, canonicalization rules, and parser requirements are defined in the v1 Spec.

The result is an artifact that degrades gracefully in both directions.

Handed to a human with no tooling, it remains a readable document whose obligations can be stated in plain language.

Handed to a capable AI system with no prior knowledge of KNOBE, it remains readable markdown, and the file can often reveal its own structure: frontmatter, body, payload markers, and explanatory text all remain visible. In practice, capable naive readers can often infer that the Base64 block is part of the object's structured record. This is discoverability, however, not verification.

Deterministic verification requires KNOBE-aware tooling. Handed to such a tool, the same file becomes a verifiable structured object with machine-legible provenance and inspectable payload fields. That distinction is central:

> *KNOBE is discoverable to capable readers and models, but verifiable only through deterministic tooling.*

Why these design choices

Each layer reflects a choice with a reason behind it:

- **Plain text**: readable by a person, a file browser, or a model, with no account, permission, or special software.
- **One file**: body and payload travel together, so there is no sidecar metadata to lose at a handoff.
- **Sealed payload**: integrity can be checked locally, by anyone, without trusting the sender or a server.
- **Declared attribution**: the record is useful now, before cryptographic identity infrastructure exists, rather than waiting for it.
- **Quarantine-first**: arrival is not endorsement; verifying an object is never the same as trusting it.
- **Non-enforcing seal**: the seal reports and never blocks, so inspection, accessibility, and adaptation stay possible.

4. Fields That Carry Obligation

The v1 payload is deliberately small. A required core names what the object is and how to read it (title, summary, content type, license, creation date, spec version, privacy and quarantine posture, and source attribution), all sealed under a payload hash. Everything beyond that core is optional structure for richer records; the exact field list, controlled vocabularies, and conformance rules are defined in the v1 Spec.

The optional fields are where KNOBE carries more than provenance. Provenance answers *where did this come from?* KNOBE also asks *how should the next party receive it?* Several optional fields carry that obligation.

`parents[]`

A `parents` entry records the KNOBEs or source objects this object derives from. Each entry links to its parent by payload hash and should carry a `title` and a `relationship`. The v1 canonical values are `adaptation_of`, `compression_of`, `synthesis_input`, `derived_from`, `responds_to`, and `supersedes`, and the vocabulary is open to namespaced extension. This makes lineage an inspectable property of the object rather than a fact lost in someone's file system.

`transformation_history[]`

A `transformation_history` entry records what changed, who changed it, when, and by what `strategy`, for example synthesis, compression, or adaptation. The Spec defines the field; strategy values are illustrative and the vocabulary remains open. A transformed object should not arrive as if it had no history.

`fidelity_limits`

A `fidelity_limits` field tells a receiver how far the object can be trusted as a representation of its source, and what must not be inferred from it. A classroom vignette may preserve the structure of a historical case while compressing chronology and inventing dialogue; a summary may preserve the main argument but not the evidentiary detail; a translation may preserve sense but not rhythm, register, or legal force. Fidelity limits are not truth guarantees. They are receiver-facing interpretive bounds.

`use_conditions`

A `use_conditions` field carries originator-declared terms the next reader or agent is asked to honor: permitted uses, disallowed uses, consent constraints, quotation constraints, required preservations, or redistribution conditions. Use conditions are declared obligations, not enforcement mechanisms. KNOBE informs; it does not control.

`accessibility`

An `accessibility` field records adaptation lineage: what was adapted, from what source (by payload hash), by whom, for what purpose, and under what review date, with an `adaptation_type` such as `caption`, `simplification`, `alt-text`, `translation`, or `multimodal`. This matters because adaptation is not a technical afterthought. Captioning, simplification, translation, alt text, audio description, and multimodal adaptation are knowledge work; they involve judgment, labor, interpretation, and fidelity limits. KNOBE gives that labor a place in the artifact's record, bound to the source it was made from.

Why these fields matter

Together, these fields make KNOBE more than a provenance wrapper. They let a knowledge object arrive carrying more of the interpretive field required to handle it responsibly, across a gap between parties who do not share the same access to it: between an author and a later reader, between a human and an agent, between a source and its adaptation. They inform; they do not enforce. A receiver is free to ignore them, but never able to say the object failed to carry them.

---

5. Integrity, Not Truth

KNOBE's stance on trust has two parts: integrity is narrow, and trust is local. This section takes the first; the next takes the second.

When a verifier shows a green check, exactly one thing has been established: the sealed payload is the same payload that was present when the artifact was sealed. The SHA-256 comparison (NIST, 2015) is a tamper-evidence mechanism, and a strong one. It is not, and cannot be, a truth mechanism.

A KNOBE can verify perfectly and still be wrong, outdated, plagiarized, misattributed, misleading, or malicious. A liar can seal a lie; the seal then faithfully proves that the lie has not been edited since. Conversely, a mismatch does not prove malice: a well-meaning edit, a line-ending conversion, or a copy-paste accident breaks a hash just as surely as tampering does.

The hash answers one question:

> *Is this the sealed payload?*

It refuses every other. KNOBE's hash is tamper-evidence, not a cryptographic signature binding the artifact to a verified identity; content-provenance standards such as C2PA (2025) provide that complementary guarantee for signed media, and v1's `identity_status: signed` anticipates such an extension without yet providing it.

This restraint is a design feature, not a limitation to be apologized for. Systems that conflate integrity with trust train their users to stop inspecting; the lock icon becomes a substitute for judgment rather than a precondition for it. KNOBE pushes the other way. Verification is cheap and automatic precisely so that human judgment can be spent where it is actually needed: on the claims, the attribution, the use conditions, the fidelity limits, and the fit between the object and its intended use. The green check is where inspection begins, not where it ends.

Language models are not verification environments

This failure has an LLM-specific variant. A language model tasked with verifying a KNOBE natively does not perform a cryptographic computation; it performs a textual prediction. Once a plausible hash-token sequence has been generated, the structural logic of a verification narrative strongly predicts that the next sequence should match and declare success. The model satisfies the narrative arc of the prompt rather than executing the operation, a textual performance of verification rather than verification itself. This is the **probabilistic verification fallacy**.

The failure was observed directly, in a documented field test during protocol development (June 2026): a language model confabulated a perfectly matched hash pair that bore no relation to the file's actual sealed payload. The recursion is what makes it instructive: in the same session, the model diagnosed the failure with precision and then committed it again in the next generation, embedding a hallucinated hash into a provenance record as if it were real. The diagnosis lived in the text; no persistent state existed to let it constrain the next token sequence. That gap (between correct reasoning about verification and the ability to enforce that reasoning across one's own next generation) is what "architectural boundary" means. A language model is not incapable of reasoning about verification; it is incapable of being the environment that performs it.

Hash verification requires a deterministic runtime. For that reason, KNOBE ships with a reference verifier, `lens.py`, not as a convenience but as that boundary. The rule is simple:

> *Models can help understand verification. Code must perform verification.*

---

6. Quarantine-First

New or external KNOBEs should be treated as quarantined until a local human or governed system marks them trusted. This is the local half of that stance, and it inverts the usual default of circulation systems, which treat arrival as implicit endorsement.

This matters because a KNOBE may arrive declaring itself `trusted`. Receiving systems should not automatically inherit that declaration; local trust is a receiving-system decision. The field `quarantine_status` records the review posture the object declares, but receiving environments should

maintain their own review layer. An external KNOBE may be valid, readable, and hash-verified while still remaining untrusted locally.

Quarantine-first has three practical consequences. First, inspection precedes action: tools that consume KNOBEs surface quarantine status prominently and do not act on quarantined build recipes or transformations without explicit human or governed approval. Second, trust becomes a recorded decision rather than an ambient assumption: someone, or some governed process, changes the status, and that decision belongs to the object's history. Third, the posture scales in both directions: a single reader applying personal judgment and an institution applying a formal review workflow use the same field, with different governance behind it.

Quarantine is not an accusation. It is the honest starting point for any object whose provenance has not yet been examined, which is every object, on arrival.

> *A verified KNOBE is not necessarily trusted. A trusted KNOBE is not necessarily true. A true claim still requires interpretation. The seal reports; the human decides.*

---

7. A System of Context, Not a System of Record

The predictable institutional objection arrives early: *we already have systems*. The objection is correct, and it misses the point.

Course systems keep grades. Ethics review keeps approvals. Repositories keep deposits. Archives keep accession records. Publishers keep formal publication metadata. Human resources systems keep employment records. Compliance systems keep official decisions. Each is a system of record, and each should remain. KNOBE replaces none of them.

KNOBE is a **system of context**. It carries object-level context *between* systems of record: attribution, source relations, transformation history, fidelity limits, use conditions, accessibility adaptations, quarantine status, and inspection cues. Systems of record are walls; objects move through the doors between them. KNOBE is the context layer that travels with the object.

Many institutions do not need another platform. They need a way for knowledge objects to move through the systems they already run without losing the context that makes them responsible to use. A KNOBE lets an object say, in a form both a person and an agent can read:

- this is what I am;
- this is where I came from;
- these are my sources;

- these are my transformations;
- these are my constraints;
- these are my fidelity limits;
- this is how I should be inspected;
- this is what has and has not been verified.

The goal is not to centralize control. It is to make movement more answerable.

---

8. AI Harnesses and Process Literacy

AI deployment now attends to the systems around models (memory, tools, permissions, retrieval, observability, execution policy) as much as to models themselves. What appears to be a model capability is often a model-plus-system capability, and performance changes materially when the same model runs under a more constrained and better-instrumented execution layer (Gu, 2026).

KNOBE does not replace that harness. It supplies a better object for any harness to handle: a knowledge artifact that arrives at any encounter already carrying its attribution, transformation history, constraints, and interpretive obligations as part of itself.

The harness controls the encounter. KNOBE preserves the object across encounters.

The harness governs execution policy, memory architecture, tool permissions, observability, and runtime constraints; KNOBE governs what the artifact carries: attribution, transformation history, fidelity limits, use conditions, and interpretive obligations. If an artifact arrives stripped of those, no downstream harness can reconstruct what was never carried; the loss compounds across hops, until agent A summarizes, agent B excerpts the summary, agent C acts on the excerpt, and no party (human or machine) can recover what the original object required responsible users to know. If the artifact arrives as a KNOBE, the harness begins with a richer, more accountable object. This is why KNOBE belongs to the emerging practice of agent harness engineering without claiming to be the harness itself: it addresses the object side of the problem.

The same machinery serves a long-standing institutional aspiration: **process literacy**, the ability to understand, document, inspect, and evaluate the pathway by which knowledge work was produced, not only its final output. Students need it to learn honestly and to show their work. Researchers need it for methods, replication, and credit. Accessibility specialists need it to record adaptations without severing them from sources. Administrators and reviewers need it to evaluate work they did not watch happen. Draft folders, chat logs, course-system submissions, ethics records, and institutional repositories each

preserve part of the picture; none reliably carries a shared, object-level account of judgment, transformation, attribution, constraint, and use condition that travels with the work. KNOBE gives the process record a portable home inside the object.

Process literacy is also where a quieter failure is fought. The **Matthew Effect**, the well-documented tendency for credit in knowledge systems to accumulate around those already prominent (Merton, 1968), is a problem of *maldistribution*: the credit exists, and it flows unfairly. The **Matthew Defect** is a more basic failure. The contribution record never survives to be credited at all. Attribution and context are stripped in transit, buried under reformatting, quietly co-opted by a more prominent name, or marginalized as unimportant, by accident, by convenience, or by design. What remains is the visible fragment; the labor, judgment, and source relations that made it meaningful are gone. Where the Effect misallocates credit that was recorded, the Defect destroys the record before allocation is even possible, and no after-the-fact redistribution can recover what was never carried. This is the human-level expression of context survivorship bias, and it is why attribution fields are protocol requirements rather than optional metadata. The requirement is deliberately workflow-neutral: it applies identically to the entirely human essay, the AI-assisted synthesis, and the archival monograph, because the point is recording what actually happened.

---

9. Adoption Without a Platform

KNOBE does not require an official authoring path. A valid v1 object may be written directly, converted from an existing document, generated from a guided form, produced through batch upload, assembled from other KNOBEs, or created inside a credentialed application environment. These are creation paths, not protocol requirements. The protocol defines the object, not the one correct way to make it.

Some KNOBEs may be designed as teaching or bootstrapping artifacts, and collections of KNOBEs may share conventions, review posture, and transformation norms. These are useful adoption patterns. A valid v1 KNOBE needs no bootstrapping artifact, guided environment, or official tool. KNOBE's own materials include such aids (teaching objects called Seeds; a guided introduction called Grove), but they are conveniences, not protocol machinery.

The public site provides current entry points for verification, authoring, examples, and implementation. Credentialed environments, institutional workflows, identity, review, and governance can be built as application layers; they are not the protocol itself.

The success of KNOBE depends on independent implementation. A developer should be able to build a compatible verifier from the specification alone, compare results against shared test vectors, and know whether the implementation conforms to v1. That is why v1 file semantics are frozen. Extensions may add optional fields; they may not reinterpret valid v1 files.

10. Limits and Invitation

It is as important to say what KNOBE does not do as what it does.

KNOBE does not solve hallucination, truth, copyright, authorship verification, identity, governance, or ethics. The hash proves integrity, never trust. Attribution and good faith are *declared* in v1, not proven; `identity_status: signed` points toward future cryptographic identity extensions, but v1 makes no such guarantee. KNOBEs do not execute: a `.knobe.md` is inert plain text, and build recipes are instructions a human or agent may choose to follow after inspection, never self-executing code. KNOBE replaces no system of record. Nor does KNOBE make AI an author: it records AI involvement as a matter of honesty and accountability, while authorship belongs to those who can answer for the work. Whether that boundary could ever move is not a question a file format settles.

KNOBE does not make knowledge immune to misuse, and it does not replace judgment, governance, ethics, or trust. It does something narrower, and for that reason useful: it makes it harder for a knowledge object to arrive stripped of the conditions needed to interpret it responsibly.

That is the invitation of v1. The specification is small enough to implement in an afternoon and precise enough to be falsifiable in the only way that matters for infrastructure, by strangers trying to build against it. Verify it. Break it. Implement it independently from the specification alone. Extend it where real work proves the schema too small. But keep the central test in view: when knowledge moves, does the object still carry enough of its knowing to be used responsibly?

This paper practices what it specifies. The document you are reading is itself a sealed KNOBE: its payload records its human authorship, its declared AI assistance, its license, its quarantine posture, and a parent receipt identifying the longer working document it compresses, by hash, with the relationship `compression_of`. It is, in other words, a compression that kept its obligations. That is the whole idea.

Appendix A. Relationship to the Normative Specification

The normative file format, required fields, canonical hash rule, body-hash behavior, verification states, conformance rules, and version semantics are defined in the KNOBE Protocol v1 Spec. This white paper is argumentative rather than normative; where this paper summarizes implementation behavior, the Spec controls. The v1.0 specification was frozen on 2026-06-21; see References for the archival citation.

References

C2PA. (2025). *C2PA Technical Specification, version 2.3*. Coalition for Content Provenance and Authenticity. https://spec.c2pa.org/specifications/specifications/2.3/

Gleick, J. (2011). *The Information: A History, a Theory, a Flood*. New York: Pantheon Books.

Gu, S. (2026). *From Model Scaling to System Scaling: Scaling the Harness in Agentic AI*. arXiv:2605.26112.

Josefsson, S. (2006). *The Base16, Base32, and Base64 Data Encodings*. RFC 4648, Internet Engineering Task Force.

Kyle, D. (2026). *KNOBE Protocol v1.0 Specification* (frozen 2026-06-21). https://knobe.org/spec. Source and test vectors: https://github.com/KnobeOne/knobe-protocol (release v1.0.0).

Mangel, M., & Samaniego, F. J. (1984). Abraham Wald's work on aircraft survivability. *Journal of the American Statistical Association, 79*(386), 259–267.

Merton, R. K. (1968). The Matthew effect in science. *Science, 159*(3810), 56–63.

National Institute of Standards and Technology. (2015). *Secure Hash Standard (SHS)*. FIPS PUB 180-4.