Trust and Proof

Robert Betts

Riley Betts Ltd
robert.betts@rileybetts.ai

7 September 2026

Abstract

Trust and proof are often spoken of as if they were one relation. Four claims get run together: that they are the same thing, that trust depends on proof, that proof depends on trust, and that proof can stand in for trust. None of them holds in every case. *Trust* is a relational attitude toward an agent, institution, method, or machine. *Proof* is used broadly here to mean an inspectable evidentiary object supporting a proposition: a formal derivation, cryptographic proof, certificate, or test result. The label "trustless" then treats the remaining demand for trust as a Boolean property of a system. This paper argues that it is not. Where trust remains can only be judged relative to a proof boundary: what has actually been proved, and relative to which assumptions and dependencies.

Write $P(x \mid A)$ for the required property of x established relative to assumptions and dependencies A, and $\mathrm{Need}(T, x \mid A)$ for the remaining demand to trust x given that boundary. Where proof successfully substitutes for trust, the substitution is local: $P(x \mid A) \rightarrow \neg\,\mathrm{Need}(T, x \mid A)$. That claim can be true. Zero-knowledge proofs, formal verification and cryptography can discharge particular trust relations. What they do not entail is the elimination of every remaining trust relation.

The trust residue $R(A)$ is the subset of A whose members still require trust relative to the rest of the boundary. The Trust Displacement Result states that local discharge does not entail global elimination:

$$P(x \mid A) \wedge \neg\,\mathrm{Need}(T, x \mid A) \;\not\Rightarrow\; R(A) = \emptyset.$$

Architectures that establish the same property of x can be compared by residue: A_2 trust-dominates A_1 when $R(A_2)$ is a proper subset of $R(A_1)$. "More trustless" then means a strictly smaller trust residue, not the absence of trust. A proof claim that does not expose its material A is an incomplete engineering claim. The Ethereum Foundation Proximity Prize illustrates the engineering consequence: disproved conjectures invalidated part of the assumed proof boundary, rather than demonstrating a failed implementation.

Governance of autonomous action is not the accumulation of logs or proof objects. It is explicit management of proof boundaries and their residues, including when A changes while the observable action does not. The interesting question is not whether a system is trustless. It is where the trust has gone.

Keywords: trust; proof; trust residue; trustless systems; formal verification; cryptographic assumptions; governance; philosophy of trust

Contents

1 Introduction

How do trust and proof relate? Is there a dependency between them, or can one stand in for the other? Four claims about that relationship are often run together:

- Trust is equivalent to proof
- Trust depends on the existence of proof
- Proof depends on the existence of trust
- Proof is a substitute for trust

The words are not interchangeable. Proof is a syntactic or evidentiary object: a derivation, a test suite, a certificate, a machine-checked artefact. Trust is an epistemic attitude toward an agent, a method, a machine, or that object. Treating them as equivalent collapses an epistemic attitude and an evidentiary object into a single relation.

The distinction matters because trust at machine scale cannot rest only on human forms of trust. Reputation, a handshake, and "I know that person" are not enough for autonomous systems making decisions across institutional and technical boundaries. The live problem is which demands for trust can be discharged by proof, which remain, and where they move when the system changes.

"Trustless" is the name usually given to the ambition of making those demands disappear. This paper makes a different claim. Trust is not usefully treated as a Boolean property of a system. Proof can genuinely remove particular requirements for trust, sometimes dramatically. But whether it has done so can only be stated relative to the assumptions and dependencies within which that proof operates.

The question is therefore not simply whether something is proved. It is what has been proved, relative to what, and which demands for trust remain after the proof has done its work.

2 Four claims, three questions

Taking each claim literally, and provisionally treating $T(x)$ and $P(x)$ as predicates over a common domain of claims, agents, artefacts, or processes, the four statements become:

Informal claim	Formal reading
Trust is equivalent to proof	$\forall x,\ T(x) \leftrightarrow P(x)$
Trust depends on the existence of proof	$\forall x,\ T(x) \rightarrow P(x)$
Proof depends on the existence of trust	$\forall x,\ P(x) \rightarrow T(x)$
Proof is a substitute for trust	$\forall x,\ P(x) \rightarrow \neg\,\mathrm{Need}(T, x)$

Two provisos travel with "provisionally." The shared domain is granted only for the argument's sake. Nothing yet licenses trusting and proving the same x, and the readings below test whether that treatment can be sustained.

$P(x)$, read as "x is proven," is already too loose. One proves a property of x, not x, and only relative to some assumptions. It is a placeholder. The argument will replace it with an assumption-relative $P(x \mid A)$, where A makes the proof boundary explicit.

$\mathrm{Need}(T, x)$ is another provisional predicate: "x requires trust to be accepted." It is introduced specifically to distinguish substitution from equivalence. Substitution does not say that trust and proof are the same thing. It says that the presence of proof can discharge a particular demand for trust.

If both dependency claims hold, we recover equivalence:

$$(\forall x,\ T(x) \rightarrow P(x)) \wedge (\forall x,\ P(x) \rightarrow T(x))$$

is, by definition:

$$\forall x,\ T(x) \leftrightarrow P(x).$$

The two conditionals are the biconditional, stated separately.

A set-theoretic variant makes the same point. Let $\mathcal{T}$ be the set of trusted things and $\mathcal{P}$ the set of proven things. Equivalence is $\mathcal{T} = \mathcal{P}$. The dependencies are the inclusions $\mathcal{T} \subseteq \mathcal{P}$ and $\mathcal{P} \subseteq \mathcal{T}$. Both inclusions hold if and only if the sets are equal. Substitution is neither equality nor inclusion. Presence in $\mathcal{P}$ is instead taken to make membership of $\mathcal{T}$ unnecessary.

The four claims therefore collapse into three questions:

1. **Equivalence.** Do T and P coincide? $\forall x,\ T(x) \leftrightarrow P(x)$
2. **Necessity.** If not, which needs which? $T(x) \to P(x)$, or $P(x) \to T(x)$, or neither
3. **Substitution.** Can P discharge a demand for T? $P(x) \to \neg\,\mathrm{Need}(T, x)$

The point of starting with these deliberately simple predicates is not that they will survive. It is to find out where they fail and what additional structure is required to state the relationship properly.

3 They are not the same

The first claim is

$$\forall x,\ T(x) \leftrightarrow P(x)$$

or, equivalently,

$$\mathcal{T} = \mathcal{P}.$$

This is not a third primitive relation. It is both necessity arrows taken together. The question still stands alone because people use the words as if they named one thing. "Trustless." "Verified means trusted." If you have one, you have the other.

That programme fails as soon as the types are kept apart.

You can hold a valid proof and still withhold trust because the proof is unread, the prover is untrusted, or the tooling is opaque:

$$P(x) \wedge \neg T(x).$$

You can also trust without proof, through a prior, a reputation, testimony, or an axiom taken without derivation:

$$T(x) \wedge \neg P(x).$$

Either case falsifies the biconditional. Together they show independence, not identity.

Wittgenstein's *On Certainty* (1969, OC §§1, 19) is enough here without yet invoking the later hinge argument. G. E. Moore, in *Proof of an External World* (1939), holds up his hands and treats "Here is one hand" as a rigorous proof. "Know" and "prove" will not do the work of the groundless certainty the practice actually turns on. Equating T with P makes the same stretch.

At machine scale the slogan is "trustless." A protocol can establish a property of a transaction while trust remains on operators, clients, compilers, specifications, implementations, and prior state. The thing proved is not the whole field of what must still be accepted.

$\forall x,\ T(x) \leftrightarrow P(x)$ is therefore a slogan of identity, not a description. Refuse to treat the words as synonyms and the next question is which, if either, depends upon the other.

4 Which needs which

The two implications are independent. Both together recover equivalence. Each alone is a different programme.

4.1 Trust depends on the existence of proof

The claim is

$$\forall x,\ T(x) \to P(x).$$

Proof is necessary for trust. There should be no $T(x)$ without $P(x)$.

This is first a norm, not a description of how belief works. W. K. Clifford, in *The Ethics of Belief* (1877), argues that it is wrong to believe on insufficient evidence. Clifford's requirement is sufficient evidence, not proof in the strict sense. The stronger programme considered here says do not grant $T(x)$ until $P(x)$.

"Don't trust, verify," read this way, says trust ought not to outrun proof. Read as replacement of trust by proof, the same slogan becomes the substitution claim considered below.

As a descriptive proposition the arrow is false. Its counterexample is

$$T(x) \wedge \neg P(x).$$

Hume, in the *Enquiry* (Sections IV–V), supplies the classical crack. We trust custom and the next case of a regularity without proving that nature will not switch. Testimony supplies another case. William James, in *The Will to Believe* (1896), argues against Clifford that some commitments cannot wait upon proof and are not therefore irresponsible.

At machine scale humans will continue placing T on autonomous systems they cannot personally P. Hardwig's epistemic dependence cuts here too. $T(x)$ may itself be received through $T(y)$, $P(y)$, or both. I may trust a system on the strength of other people's proofs, instruments and testimony without holding those proofs myself.

As a norm, the demand that trust should not outrun proof becomes more urgent at machine scale. As a universal description, $\forall x,\ T(x) \to P(x)$ is false.

4.2 Proof depends on the existence of trust

Now reverse the arrow:

$$\forall x,\ P(x) \to T(x).$$

Trust is necessary for proof.

Taken literally, this says the proven thing is itself trusted. As a universal that is also false. We can possess $P(x)$ while withholding $T(x)$, perhaps because the proof has not been inspected, the prover is not accepted, or the path from formal artefact to running system is in doubt. The proposition survives only by trivialising into "we trust what we have just proved," which begs the question it was raised to answer.

The serious reading of "proof depends on trust" is not

$$P(x) \to T(x)$$

but a dependency across different objects. $P(x)$ is received through assumptions and dependencies concerning y, z, and others.

John Hardwig, in *Epistemic Dependence* (1985) and *The Role of Trust in Knowledge* (1991), makes this dependence explicit. The argument and data available to one knower often arrive through others. Steven Shapin, in *A Social History of Truth* (1994), and Michael Polanyi, in *Personal Knowledge* (1958), make related cuts. Experimental proof operates within communities of credit, and no intelligence operates outside what Polanyi calls a fiduciary framework. Wittgenstein's hinges in *On Certainty* (§§341–343) put the logical point more starkly: proofs turn on commitments that are not themselves presently being proved.

Machine proof makes the boundary easier to see.

A machine-checked proof can extend through specification, implementation, compiler, kernel, and even a model of the hardware. Formal methods can move that boundary dramatically. They do not, merely by proving another layer, make the idea of a boundary disappear.

The unary predicate $P(x)$ has now reached its limit. What is actually established is closer to

$$P(x \mid A)$$

where A identifies the assumptions and dependencies against which the proof is established or on which its acceptance depends.

This is not yet a claim that every member of A must be trusted. Some members of A may themselves be proved. Others may be eliminated by architecture, cryptography, replication, formal verification, or a different construction. Others may remain outside the proof boundary.

The distinction matters. The presence of A does not show that proof is futile. It gives us a way to ask exactly what proof has achieved.

5 Proof relative to a boundary

The literature already in view does not say that proof is worthless. It says that a proof is never a free-standing object. What follows is only the notation needed to keep that fact in the formulae.

Wittgenstein's hinges (OC §§341–343) are the logical reason. A proof turns on commitments that are not, in the same movement, being proved. Those commitments are not a defect in the proof. They are the conditions under which the proof can be a proof at all. Write them as a set A, and "x is proven" becomes "the required property of x is established relative to A."

Hardwig, Shapin and Polanyi fill A from the other side. The argument and the data arrive through others (Hardwig, 1991). To test the next instrument is only to enter a further chain of credit (Shapin, 1994). No intelligence operates outside a fiduciary framework (Polanyi, 1958, pp. 266–267). A is therefore not an optional list of caveats. It is the dependence through which P is received.

Szabo (2001) is the engineering form of the same cut. Trusted-third-party assumptions are the principal source of cost and risk in a security protocol. The design goal is to minimise them and plug what remains with mechanism. He does not claim that proof discharges every demand for trust. In the spirit of his argument, no protocol is fully free of such assumptions. In the terms used here, mechanism can shrink A. It does not license dropping A from the claim.

Two building blocks follow. First, make the boundary of the proof explicit: $P(x \mid A)$. Second, make the remaining demand for trust explicit relative to that same boundary: $\mathrm{Need}(T, x \mid A)$. The coarse substitution slogan $P(x) \to \neg\,\mathrm{Need}(T, x)$ treats A as empty or irrelevant. The local claim $P(x \mid A) \to \neg\,\mathrm{Need}(T, x \mid A)$ asks only whether this proof, given these assumptions, discharges this demand. That is the question the literature permits, and the one the next definitions state.

Let

$$P(x \mid A)$$

mean that the required property of x is established relative to a set of assumptions and dependencies A.

Correspondingly, replace the earlier provisional predicate $\mathrm{Need}(T, x)$ with

$$\mathrm{Need}(T, x \mid A)$$

meaning that, given A, acceptance of x still requires trust in x.

This changes the substitution question.

The earlier universal proposition

$$P(x) \to \neg\,\mathrm{Need}(T, x)$$

was too coarse. The useful proposition is local:

$$P(x \mid A) \to \neg\,\mathrm{Need}(T, x \mid A)$$

and unlike the earlier claims, this can be true.

Proof can genuinely discharge trust.

Suppose Alice asserts that she knows a witness w satisfying some statement x. Bob could accept Alice's assertion because he trusts Alice. But if Alice supplies a sound zero-knowledge proof, Bob need no longer trust her honesty about possession of the witness. Relative to the proof system and its assumptions A, verification has discharged that particular requirement for trust.

This is not trust moved by wordplay. It is trust removed from a specific relation.

Formal verification can do the same. If a programmer claims that an implementation preserves an invariant, a mechanically checked derivation can replace the need to trust that assertion. Cryptographic signatures can replace trust in the asserted origin of a message with verification under cryptographic and key-management assumptions. Consensus protocols can remove particular dependencies on individual participants.

The substitution programme therefore succeeds locally.

The remaining question is whether local substitution entails global elimination.

It does not.

6 Trust residue

Local substitution answers one question: given A, has this proof discharged the demand to trust x? It does not answer another: of the members of A themselves, which still require trust?

That second question is forced by the same literature. Wittgenstein's hinges remain outside the present justification. Hardwig's dependence and Shapin's chain of credit do not terminate when one claim is proved. Szabo's leftover third-party assumptions are the engineering case of the same remainder. The set A is therefore not homogeneous. Some of its members may themselves be discharged by proof. Others continue to require trust. The trust residue is that remainder, named so that local success cannot be mistaken for an empty system.

For a particular proof claim, let

$$A = \{a_1, a_2, \ldots, a_n\}$$

be the assumptions and dependencies associated with establishing or accepting $P(x \mid A)$. Some members of A may themselves be discharged by proof. Others may continue to require trust. Define the **trust residue** of the proof boundary as

$$R(A) = \{a \in A : \mathrm{Need}(T, a \mid A \setminus \{a\})\}.$$

$R(A)$ is the part of the boundary that proof does not discharge. It must be trusted for $P(x \mid A)$ to be accepted. A is individuated at the level of atomic trust requirements. A "one-of-n honest" or "k-of-n honest" condition is a single member of A, not n separate members. A disjunctive trust assumption therefore appears as one non-dischargeable element, and the residue stays a flat set.

Some boundaries have circular discharge dependencies. There a is provable only if b is trusted, and b is provable only if a is trusted, so trusting either suffices but neither is naturally atomic. A flat set cannot capture this. The residue must then be read as a family of minimal trusted sets. We assume acyclic discharge, under which the flat residue is well-defined.

This separates two claims that "trustless" tends to collapse.

Local trust discharge. The proof removes the demand to trust x itself, given the boundary:

$$P(x \mid A) \to \neg\, \mathrm{Need}(T, x \mid A).$$

Global trust elimination. Nothing in the boundary is left to trust:

$$R(A) = \emptyset.$$

The first does not entail the second:

$$P(x \mid A) \wedge \neg \mathrm{Need}(T, x \mid A) \;\not\Rightarrow\; R(A) = \emptyset.$$

Call this the **Trust Displacement Result**. A proof may discharge the requirement to trust x while leaving a non-empty trust residue among the assumptions and dependencies required to establish or accept that proof.

This is a deliberately limited result. It does not assert that $R(A)$ must always be non-empty; there are boundaries whose dependencies are, in fact, all dischargeable by proof. Nor does it say that every member of A is a matter of trust. The claim is only the non-entailment: from the discharge of the demand to trust x, an empty residue does not follow.

From the fact that proof has eliminated a particular demand for trust, it does not follow that trust has been eliminated from the containing system.

The distinction is

$$\begin{aligned} &\text{proof of } x \\ &\neq \text{trust discharged for } x \\ &\neq \text{trust eliminated from the system.} \end{aligned}$$

The philosophical arguments considered earlier now have a more precise role. Hardwig, Shapin and Polanyi explain why trust residues arise in epistemic practice. Wittgenstein explains why justification operates against commitments not simultaneously placed under justification. They need not establish the formal result. They explain why its residual case matters.

7 Moving the boundary

If global elimination does not follow from local discharge, the remaining engineering question is not whether a system is trustless. It is whether one construction leaves a smaller, or better placed, residue than another for the same property of x.

That is Szabo's programme once A is kept in the claim. Trusted-third-party assumptions are to be minimised and plugged with mechanism. The literature already used does not forbid that movement. Wittgenstein's hinges can shift; Hardwig's dependence can be reassigned; Shapin's chain of credit can be shortened at one link and lengthened at another. What it forbids is treating the movement as disappearance.

Once the residue is explicit, two boundaries that establish the same property of x can be compared by the trust they leave. Suppose $P(x \mid A_1)$ and $P(x \mid A_2)$ both establish that property, with residues $R(A_1)$ and $R(A_2)$. Say A_2 **trust-dominates** A_1 for x, written

$$A_2 \prec_t A_1,$$

when

$$R(A_2) \subsetneq R(A_1).$$

The second construction discharges at least one trust dependency the first retained, and introduces none the first did not have. "More trustless" then means a strictly smaller residue for the same property. It does not mean the absence of trust.

$\prec_t$ is a partial order, and a deliberately weak one. It compares boundaries over a shared universe of assumptions. That is the case where A_2 is obtained from A_1 by discharging or relocating specific dependencies. Two architectures built on genuinely different assumptions need not be comparable at all. Their residues are drawn from different universes. Set inclusion between them is undefined without a semantics that maps one assumption onto another, or a

cost function that weighs residual dependencies by consequence. That weighting is real and unavoidable. One privileged operator can outweigh three independently verifiable components, so cardinality is never the measure. It is left to a later treatment. What $\prec_t$ fixes now is the shape of the comparison. The movement of trust across a boundary can be stated, rather than hidden behind a Boolean label.

This also gives the substitution programme its strongest form.

Proof, cryptography and formal verification can remove trust dependencies. Better systems should do exactly that. The error lies not in attempting substitution, but in inferring from successful local substitution that the system itself has become trustless.

8 Related work

That a security guarantee is stated relative to assumptions is not new. Several disciplines already name parts of the residue. The trusted computing base isolates the components whose correctness is assumed rather than checked. Threat modelling enumerates the adversary and environment assumptions a guarantee is conditioned on. Trusted setup names the ceremony a cryptographic scheme must trust. In distributed systems and multiparty computation the residue is usually written as an honest-fraction condition, a t-of-n or one-of-n-honest assumption, and protocols are compared by whose honesty they require. Blockchain practice has made that comparison explicit. Layer-two "trust assumption" taxonomies and risk frameworks classify systems by which parties must be trusted for safety and for liveness, and "minimal trust" is a stated design goal.

$P(x \mid A)$ and the residue $R(A)$ are not offered in place of any of these. They are the common form beneath them. On one side they join the epistemology of testimony: Hardwig's dependence, Shapin's chains of credit, Wittgenstein's hinges. On the other they join the engineering practice. The residue $R(A)$ is the object those disciplines already compute case by case. Naming it, and giving it an order $\prec_t$, is what lets "more trustless" be compared across designs rather than asserted of one. The contribution is the unification and the ordering, not the discovery that assumptions exist.

9 A proof claim includes its boundary

The notation $P(x \mid A)$ is not a warning label attached after the fact. It is the shape of the claim. If A is left implicit, the operational claim is read as $P(x)$. When a member of A later fails, the failure is then misdescribed as a broken implementation rather than a broken, or incomplete, boundary.

That is why exposing A is an engineering requirement, not a philosophical courtesy. Szabo's leftover assumptions, Wittgenstein's hinges, and Hardwig's received argument are all members of the claim.

A recent cryptographic episode makes the cost of omitting A concrete. It shows the mechanism, not a mere failure. Hash-based SNARKs — FRI (Ben-Sasson et al., 2018), DEEP-FRI (Ben-Sasson et al., 2020), STIR (Arnon et al., 2024a) and WHIR (Arnon et al., 2024b) — rest their soundness arguments on Reed–Solomon proximity-gap conjectures, and in particular on aggressive "up-to-capacity" forms of them used to justify efficient parameters (Ben-Sasson et al., 2023). In 2025 the Ethereum Foundation's Proximity Prize offered roughly one million dollars to settle the up-to-capacity conjecture. Within a single week those conjectures were disproved independently by three groups: Diamond and Gruen (2025); Crites and Stewart (2025); and Ben-Sasson, Carmon, Haböck, Kopparty and Saraf (2025). Minimal repaired versions were proposed below the capacity bound. The prize remains open for related questions.

The instructive part is what did not happen. No deployed system was shown to be broken. The disproof invalidated a region of the parameter space that had been treated as safe. Moving

to parameters with proven soundness costs roughly a factor of two in proof size and verifier time, and leaves prover time largely untouched (Mohnblatt, 2025). It would therefore be wrong to read the episode as "the implementation was incorrect." The precise description is that the operational security claim had a boundary.

Let x be the claimed soundness property and let

$$A = \{a_1, \ldots, c, \ldots, a_n\}$$

where c is a conjecture required by the argument.

Then the operational claim was not an unconditional

$$P(x)$$

but

$$P(x \mid A).$$

Disproving c does not establish that every possible construction of x is false. It establishes that this particular A cannot support the claimed guarantee in the way previously assumed. The boundary moved, and its cost became visible, exactly where a member of A had been left implicit.

That difference matters. The case does not prove that assumptions are ineliminable. It shows that assumptions are part of the claim.

This yields a practical consequence:

> **A proof claim that does not expose its material A is an incomplete engineering claim.**

The proof object tells us what follows. The boundary tells us what must hold for that conclusion to carry the meaning attributed to it.

In security engineering this is familiar in the language of trusted computing bases, threat models, trusted setups, and cryptographic assumptions. The point here is not to rediscover those disciplines. It is to connect them to the broader relation between trust and proof and give the movement of trust across their boundaries an explicit form.

10 Trust, reliance and control

The formal result does not make the philosophical distinctions redundant. It sharpens where they apply.

Onora O'Neill's *A Question of Trust* (2002) shows what happens institutionally when transparency, audit and performance measures are treated as if producing more evidence automatically produces more trust. They can instead feed a culture of suspicion. The mistake is not verification itself. It is failing to distinguish the trust dependency verification actually removes from those it leaves untouched or creates elsewhere.

Annette Baier, in *Trust and Antitrust* (1986), makes another distinction. Trust involves accepted vulnerability and discretionary power. Contracts, surveillance and complete verification can reduce that vulnerability, but as they do so the resulting relationship increasingly becomes one of control or reliance rather than trust.

This gives local substitution a limit of a different kind. If $P(x \mid A)$ completely discharges $\mathrm{Need}(T, x \mid A)$, the result is not necessarily "better trust." In Baier's terms, the relation may no longer be one of trust at all.

Trust, as she draws it, is not any dependence. It is accepted vulnerability to another's discretionary power, typically a reliance on good will (Baier, 1986, pp. 234–236). Proof, contract and surveillance belong to the opposing "antitrust" picture: they try to cancel that vulnerability

by constraint. Where they succeed for a particular x, the other party is no longer trusted about x. They are verified, or they are controlled. Local substitution can therefore be a genuine engineering success and still fail as a recipe for trust, because the thing obtained is a different relation.

Jacques Derrida's treatment of testimony (1998) makes a related point. Testimony structurally asks for belief. If an event can instead be demonstrated as an object of knowledge, testimony is no longer doing the epistemic work. Proof has not perfected testimony. It has displaced it.

Niklas Luhmann's account of trust as a mechanism for reducing social complexity under incomplete information (1968/1979) points in the same direction. Proof, law, calculation and trust can all reduce complexity, but they do so differently.

The substitution programme is therefore neither simply false nor simply successful.

Proof can eliminate a demand for trust. Where it succeeds completely with respect to that demand, the resulting relation may cease to be trust. And whether the containing system has thereby reduced its overall dependence on trust is a separate question answered only by examining the new boundary and its residue.

11 Governance as management of the proof boundary

The previous distinctions now have an operational use. O'Neill's warning is that accumulating proof-tokens, logs and audits can be mistaken for trustworthiness. Baier's is that tightening verification can replace a trust-relation with control without that fact being noticed. Both become concrete once an action is authorised by a machine.

Governance, in the terms already used, is not a second proof of x. It is the work of naming A, discharging what can be discharged, and remaining answerable for $R(A)$. That work does not wait for a human sign-off on every act. It is management of the proof boundary as the system acts.

An autonomous transfer makes the cut visible.

Consider an autonomous agent authorised to initiate a transfer:

$$x = \text{“the agent may transfer £10,000 from account } \alpha \text{ to account } \beta\text{.”}$$

Suppose a governance mechanism establishes $P(x \mid A)$, where A contains at least:

- a_1: the policy specification is authoritative
- a_2: the identities of α and β are correct
- a_3: the balance and account state supplied to the decision are correct
- a_4: the policy evaluator implements the specification correctly
- a_5: execution cannot bypass the evaluator

Without further structure, the organisation may simply say that the transfer was “verified.”

But verification of what?

Formal verification of the policy evaluator might discharge the trust dependency represented by a_4.

Cryptographic identity and authorisation could discharge much of a_2.

An architecture in which every execution path is mechanically forced through the policy evaluator could discharge a_5.

Neither result proves a_1 or a_3.

The authoritative policy may still be an institutional fact determined by people and governance processes. Account state may arrive from an external ledger or system whose correctness is outside the present proof boundary.

After verification, therefore, we might have

$$R(A) = \{a_1, a_3\}.$$

The exact residue will depend upon the system. The point is that it can be named.

A second architecture might prove the integrity of the state transition from an independently verified ledger, discharging a_3:

$$R(A') = \{a_1\}.$$

If both architectures establish the same required property x, then

$$A' \prec_t A$$

relative to x.

That is a meaningful governance improvement. The organisation has not made the system metaphysically trustless. It has removed a specific technical trust dependency while leaving the institutional source of policy authority explicit.

This is where the distinction between observability and governance becomes important. Logging the transfer tells us what happened. A proof artefact may establish a property of what happened. Neither, by itself, identifies the assumptions and dependencies on which acceptance of that proof rests.

Governance is therefore not simply the accumulation of evidence, audit logs or proof objects.

> **Governance is the explicit management of proof boundaries and their trust residues.**

For an autonomous system, that means identifying A, determining which members of A can be discharged mechanically, exposing the residue $R(A)$, and ensuring that changes to the system do not silently change either.

A software update, model replacement, policy change, new data source or new execution route can change A even where the externally visible action x remains the same. The governance problem is consequently dynamic. It is not enough to prove x once. The proof boundary itself is part of the governed state.

12 Where the trust goes

The opening asked how trust and proof relate, and whether one can stand in for the other. The four claims treated that as a single relation holding of x itself. The rest of the argument has been to show why that is the wrong grain. Hardwig's received proof, Wittgenstein's hinges, Szabo's leftover assumptions, and O'Neill's proof-tokens all ask a more local question: not whether trust has vanished, but where it now sits.

What remains is to put the four claims back in that grain, and to replace the Boolean slogan with the question the title already names.

The four claims at the beginning were deliberately simple:

$$\begin{aligned}
&\forall x,\ T(x) \leftrightarrow P(x)\\
&\forall x,\ T(x) \rightarrow P(x)\\
&\forall x,\ P(x) \rightarrow T(x)\\
&\forall x,\ P(x) \rightarrow \neg\,\mathrm{Need}(T, x).
\end{aligned}$$

None adequately describes the relation.

Trust and proof are not identical. Neither simple dependency arrow holds universally. Proof can substitute for trust, but the substitution is local to a property and a boundary.

The more useful objects are $P(x \mid A)$ and $\mathrm{Need}(T, x \mid A)$. From them we can distinguish the trust discharged for x from the trust residue

$$R(A) = \{a \in A : \mathrm{Need}(T, a \mid A \setminus \{a\})\}$$

and state the central result:

$$P(x \mid A) \wedge \neg \mathrm{Need}(T, x \mid A) \;\not\Rightarrow\; R(A) = \emptyset.$$

Proof can therefore move the trust boundary, sometimes dramatically. Formal verification can replace trust in a programmer's assertion with a mechanically checked derivation. Zero-knowledge proof can remove the need to trust a prover's honesty about a witness. Cryptography can remove particular counterparty dependencies. Better architecture can reduce the trusted computing base.

Those are real eliminations of trust.

What does not follow is that because one demand for trust has been discharged, the containing system has become trustless.

A meaningful claim about proof must therefore expose the boundary within which it holds. A meaningful claim about trust reduction must expose the residue left by moving that boundary.

"Trustless" is not usefully a Boolean property of a system. Trust reduction is a relation between proof boundaries and the residues they leave behind.

The interesting question is not whether a system is trustless. It is where the trust has gone.

13 References

- Arnon, G., Chiesa, A., Fenzi, G. and Yogev, E. (2024a) 'STIR: Reed–Solomon proximity testing with fewer queries', in Reyzin, L. and Stebila, D. (eds.) *Advances in Cryptology – CRYPTO 2024*. Cham: Springer, pp. 380–413.

- Arnon, G., Chiesa, A., Fenzi, G. and Yogev, E. (2024b) 'WHIR: Reed–Solomon proximity testing with super-fast verification', *Cryptology ePrint Archive*, Paper 2024/1586. Available at: `https://eprint.iacr.org/2024/1586` (Accessed: 7 September 2026).

- Baier, A. (1986) 'Trust and antitrust', *Ethics*, 96(2), pp. 231–260.

- Ben-Sasson, E., Bentov, I., Horesh, Y. and Riabzev, M. (2018) 'Fast Reed–Solomon interactive oracle proofs of proximity', in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Leibniz International Proceedings in Informatics, 107. Dagstuhl: Schloss Dagstuhl, article 14.

- Ben-Sasson, E., Goldberg, L., Kopparty, S. and Saraf, S. (2020) 'DEEP-FRI: Sampling outside the box improves soundness', in *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Leibniz International Proceedings in Informatics, 151. Dagstuhl: Schloss Dagstuhl, article 5.

- Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S. and Saraf, S. (2023) 'Proximity gaps for Reed–Solomon codes', *Journal of the ACM*, 70(5), Article 33.

- Ben-Sasson, E., Carmon, D., Haböck, U., Kopparty, S. and Saraf, S. (2025) 'On proximity gaps for Reed–Solomon codes', *Cryptology ePrint Archive*, Paper 2025/2055. Available at: `https://eprint.iacr.org/2025/2055` (Accessed: 7 September 2026).

- Clifford, W.K. (1877) 'The ethics of belief', *Contemporary Review*, 29, pp. 289–309.

- Crites, E. and Stewart, A. (2025) 'On Reed–Solomon proximity gaps conjectures', *Cryptology ePrint Archive*, Paper 2025/2046. Available at: `https://eprint.iacr.org/2025/2046` (Accessed: 7 September 2026).
- Derrida, J. (1998) *Demeure: Maurice Blanchot*. Paris: Galilée. English edition: Blanchot, M. and Derrida, J. (2000) *The Instant of My Death / Demeure: Fiction and Testimony*. Translated by E. Rottenberg. Stanford, CA: Stanford University Press.
- Diamond, B.E. and Gruen, A. (2025) 'On the distribution of the distances of random words', *Cryptology ePrint Archive*, Paper 2025/2010. Subsequently published in Heninger, N. and Rosulek, M. (eds.) (2026) *Advances in Cryptology – CRYPTO 2026*. Lecture Notes in Computer Science, 16803. Cham: Springer, pp. 96–127. Available at: `https://eprint.iacr.org/2025/2010` (Accessed: 7 September 2026).
- Ethereum Foundation (2025) *The Proximity Prize*. Available at: `https://proximityprize.org/` (Accessed: 7 September 2026).
- Hardwig, J. (1985) 'Epistemic dependence', *The Journal of Philosophy*, 82(7), pp. 335–349.
- Hardwig, J. (1991) 'The role of trust in knowledge', *The Journal of Philosophy*, 88(12), pp. 693–708.
- Hume, D. (1748) *Philosophical Essays Concerning Human Understanding*. London: A. Millar. Later issued as *An Enquiry Concerning Human Understanding*.
- James, W. (1896) 'The will to believe', *The New World*, 5, pp. 327–347.
- Luhmann, N. (1968) *Vertrauen: Ein Mechanismus der Reduktion sozialer Komplexität*. Stuttgart: Ferdinand Enke.
- Luhmann, N. (1979) *Trust and Power*. Translated by H. Davis, J. Raffan and K. Rooney. Chichester: John Wiley & Sons. English translation of the 1973 enlarged edition of *Vertrauen* and of *Macht* (1975).
- Mohnblatt, N. (2025) 'Proximity gaps: what happened and how does it affect our SNARKs'. Available at: `https://blog.zksecurity.xyz/posts/proximity-conjecture/` (Accessed: 7 September 2026).
- Moore, G.E. (1939) 'Proof of an external world', *Proceedings of the British Academy*, 25, pp. 273–300.
- O'Neill, O. (2002) *A Question of Trust: The BBC Reith Lectures 2002*. Cambridge: Cambridge University Press.
- Polanyi, M. (1958) *Personal Knowledge: Towards a Post-Critical Philosophy*. London: Routledge & Kegan Paul.
- Shapin, S. (1994) *A Social History of Truth: Civility and Science in Seventeenth-Century England*. Chicago: University of Chicago Press.
- Szabo, N. (2001) 'Trusted third parties are security holes'. Originally published 2001. Available at: `https://nakamotoinstitute.org/library/trusted-third-parties/` (Accessed: 7 September 2026).
- Wittgenstein, L. (1969) *On Certainty*. Edited by G.E.M. Anscombe and G.H. von Wright. Translated by D. Paul and G.E.M. Anscombe. Oxford: Basil Blackwell.