Admissibility Certificates

Trust as a Property of System Histories

Flyxion
Independent Researcher

July 2026

Abstract

Trust in an adaptive system is usually treated as a property of its current state: a model is secure if its present weights, outputs, or behavior satisfy some check. This paper argues that this framing is incomplete, and in adversarial or continuously-adapting settings, actively misleading. Two systems can occupy an identical present state while differing entirely in how they arrived there — one through authenticated data, reproducible experimentation, and documented governance; the other through undocumented intervention, silent drift, or adversarial manipulation. The states are indistinguishable. The histories are not, and only one is admissible.

We formalize this distinction. A system is represented not as a static configuration but as a history $H_\Sigma = \{\Sigma(t) : t \in I\}$, where $\Sigma(t)$ includes parameters, data, lineage, pipeline state, governance, and recovery capacity at time t. Admissibility, $\mathcal{A}(H_\Sigma)$, is defined over the history rather than any single point within it. This motivates the paper's central formal object, the admissibility certificate C_Σ: a structured record of provenance, reconstruction information, monitoring predicates, and recovery operators that certifies not a state but the trajectory that produced it.

We ground this framework empirically through Susanna Cox's *Securing AIML Systems in the Age of Information Warfare*, an industrially-motivated MLOps security framework that independently arrives at nearly every component of C_Σ (trusted baselines, monitoring thresholds, failure-mode documentation, lineage tracking, and incident recovery) without recognizing them as instances of a single underlying structure. Reading Cox's framework through this lens also exposes a distinction largely absent from the security and MLOps literature: rollback restores an artifact, reconstruction restores a state, and only repair restores admissibility to the history itself — a strict hierarchy, Rollback $\subsetneq$ Reconstruction $\subsetneq$ Repair, with consequences for when retraining or recovery procedures actually resolve a security incident rather than merely concealing it.

The paper's central theorem, the *History Principle*, generalizes the construction: any adaptive system whose future evolution depends on accumulated information requires admissibility to be defined over histories rather than instantaneous states. This applies well beyond machine learning — to operating systems, infrastructure, biological evolution, scientific reproducibility, governance, chains of custody, and distributed consensus — suggesting that AI security, rather than being the paper's subject, is one motivating instance of a more general mathematics of maintaining systems within admissible regions of history-space.

1 Admissibility Beyond Geometry

The term *admissibility* most often appears in geometric and optimization contexts: an admissible point satisfies a system of constraints, an admissible control lies within a feasible region, an admissible trajectory respects boundary conditions. In each of these settings, admissibility is a predicate on a single mathematical object — a point, a control input, a path through a fixed space — evaluated against a fixed set of rules known in advance.

Geometric admissibility is one important special case of a broader notion: admissibility as a constraint on which system histories are permitted to occur. Geometric admissibility is the special case in which the system in question is static, the constraints are known in advance, and the space of possible histories collapses to a space of possible positions.

Consider what changes when the object being evaluated is not a point but a process. A bridge is not merely required to occupy an admissible configuration under load; it is required to have been built according to admissible procedures, from materials with verifiable provenance, inspected at admissible intervals, and repaired — when repair was necessary — in ways that restore rather than merely conceal degradation. An admissible bridge, in other words, is not a bridge whose current cross-section satisfies a stress equation. It is a bridge whose history satisfies a much larger set of constraints, of which the present cross-section is only the final, visible trace.

The same is true of adaptive information systems generally, and of AI systems in particular. A deployed model's present weights θ_t may satisfy every accuracy, fairness, and robustness benchmark an organization can devise, and still be inadmissible: if those weights were reached through a training process contaminated by poisoned data, undocumented interventions, or manipulated feedback, the check that examines only θ_t cannot see this. It examines a point. The compromise lives in the history.

This calls for a reformulation. Rather than asking whether a static configuration x belongs to an admissible set $X_{\text{admissible}}$, we ask whether the history H that produced the current configuration belongs to an admissible set of histories, a notion made precise in Section 2. The remainder of this paper develops that reformulation formally. It is already implicit, though unnamed, in industrial AI security practice, and it generalizes far beyond AI to any system whose present trustworthiness cannot be separated from the process that produced it.

Before turning to that generalization, however, we must be precise about what a history is, and what it means for a predicate to be defined over one rather than over a state. That is the task of Section 2.

2 Systems as Histories

To make the claim of Section 1 precise, we need an object richer than a state and a mechanism for evaluating admissibility over that object rather than over any single instant.

System state. Let a system at time t be described by

$$\Sigma(t) = (\theta_t, D_t, L_t, P_t, G_t, R_t, \ldots), \tag{1}$$

where θ_t denotes the system's operative parameters (model weights, configuration, or equivalent), D_t the data it currently depends on, L_t its lineage — the record of how D_t and θ_t were produced — P_t the pipeline or process currently governing its evolution, G_t its governance state (who may act on it, under what authority), and R_t its recovery capacity: the resources and procedures available should the system need to be restored to an earlier admissible configuration. This tuple is deliberately open-ended; the point is not to fix its contents once and for all, but to insist that $\Sigma(t)$ is never merely θ_t. A system that has been reduced to its parameters has already had its history discarded.

System history. Define the history of the system over an interval I as the ordered trajectory

$$H_\Sigma : I \to \mathcal{S}, \qquad t \mapsto \Sigma(t), \qquad \text{equivalently } H_\Sigma = (\Sigma(t))_{t \in I}. \tag{2}$$

H_Σ is deliberately not written as a set of states. A set discards ordering, and this paper's argument depends throughout on order and temporal dependence: the sequence in which data was acquired, the order in which interventions occurred, which transitions preceded which. H_Σ is not a sequence of unrelated snapshots. It is the trajectory itself — the full record of how the system moved through state-space, including not only where it has been but the order, rate, and manner in which it got there.

The identical-endpoint problem. The reason this distinction matters, and not merely as a formal nicety, is that state alone underdetermines history. It is entirely possible for two systems to satisfy $\Sigma_1(T) = \Sigma_2(T)$ at some time T — identical weights, identical present data, identical measured behavior — while $H_{\Sigma_1} \neq H_{\Sigma_2}$.

Suppose Σ_1 reached $\Sigma_1(T)$ through a sequence of authenticated data acquisitions, reproducible training runs, documented feature engineering, and audited retraining events. Suppose Σ_2 reached the numerically identical $\Sigma_2(T)$ through a training run partially contaminated by a data-poisoning attack, followed by an undocumented manual weight edit intended to mask the resulting behavioral anomaly on the specific benchmarks an auditor was known to check.

No inspection of $\Sigma_1(T)$ or $\Sigma_2(T)$ alone — no test suite, no benchmark, no static analysis — can distinguish these systems, because by construction their present states are equal. Any property defined purely as a predicate on $\Sigma(T)$ is blind to exactly the distinction that matters. Only one of these histories is admissible. The other has produced an object that merely resembles an admissible one. This situation recurs often enough throughout the paper to deserve a name.

Definition 1 (Identical-Endpoint Problem)**.** *Two systems Σ_1, Σ_2 exhibit the identical-endpoint problem at time T if $\Sigma_1(T) = \Sigma_2(T)$ while $H_{\Sigma_1} \neq H_{\Sigma_2}$.*

This is not a pathological edge case constructed for the sake of argument. It is close to the defining structure of a sophisticated poisoning or backdoor attack: the attacker's objective is typically to reach a state that is behaviorally indistinguishable from an admissible one under ordinary inspection, while differing in some narrow, triggerable respect the defender has not thought to check. State-level admissibility checks are, by construction, the wrong tool for this class of threat. History-level admissibility checks are not, because they do not ask only what the system currently does — they ask how it came to do it.

Admissibility as a predicate on histories. We therefore define admissibility not as membership of $\Sigma(t)$ in a set of admissible states, but as a predicate evaluated on the entire trajectory,

$$\mathcal{A}(H_\Sigma) \in \{0, 1\}. \tag{3}$$

It is convenient to name the set this predicate picks out. Write

$$\mathcal{H}_{\text{adm}} = \{H_\Sigma : \mathcal{A}(H_\Sigma) = 1\} \tag{4}$$

for the space of admissible histories; later sections refer to $\mathcal{H}_{\text{adm}}$ directly rather than restating the predicate in full each time. A system is admissible at time T not when its current configuration passes inspection, but when the full history that produced that configuration — its data provenance, the integrity of its intermediate states, the documentation of its transitions, and the availability of a recovery path should any of this fail to hold — satisfies the constraints the organization, discipline, or domain requires.

This immediately clarifies what documentation, monitoring, and lineage-tracking are actually for. They are not auxiliary compliance artifacts sitting alongside the "real" system. They are the empirical evidence by which H_Σ — an object that is not directly observable in full, since most of it lies in the past — is reconstructed and evaluated. A system with no lineage records, no monitoring history, and no documented provenance is not necessarily inadmissible; but its admissibility is unverifiable, which for practical purposes amounts to the same thing. Trust cannot attach to a history no one can see.

Toward certificates. This raises an immediate practical question: if admissibility depends on a history that is neither fully observable nor exhaustively recorded, what object could possibly serve as evidence for $\mathcal{A}(H_\Sigma)$? The answer is what we call an *admissibility certificate*, introduced formally in Section 7 once its necessity has been demonstrated further. Sections 3 through 6 build the machinery — positioning the framework against neighboring fields, grounding it in an industrial case study, and developing the reconstruction/repair and reachability apparatus — that the certificate will ultimately need to carry.

Notation. For reference, Table 1 summarizes the symbols introduced in this paper. Each is also defined in prose at its first use; the table is a convenience for returning to a later section without re-reading from the start.

3 Related Work: Three Neighboring Fields

The formal machinery developed in this paper draws on, and could be mistaken for, three existing bodies of work. Each is acknowledged briefly here, not to survey them exhaustively, but to make clear what this paper takes from each and where it departs.

Reachability analysis. Control theory and formal verification have long studied reachability: given a system's dynamics and a set of initial states, which states can be reached, and can an unsafe region be shown unreachable under all admissible controls. The trajectory-based reformulation of Section 6 borrows this geometry directly — an attack, in this paper's

Symbol	Meaning
$\Sigma(t)$	System state at time t: $(\theta_t, D_t, L_t, P_t, G_t, R_t, \ldots)$
θ_t	Operative parameters (weights, configuration)
D_t	Data the system currently depends on
L_t	Lineage — how D_t and θ_t were produced
G_t	Governance state — who may act, under what authority
R_t	Recovery capacity — resources to restore an earlier admissible state
H_Σ	System history: $(\Sigma(t))_{t\in I}$, the full ordered trajectory
$\mathcal{A}(\cdot)$	Admissibility predicate, evaluated over a history
C_Σ	Admissibility certificate attached to Σ
$\mathcal{P}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{B}, \mathcal{Q}, \mathcal{R}$	One realization of C_Σ's coordinates (Section 7)
$H(C_\Sigma)$	The certificate's own history
Γ	Space of all possible trajectories through history-space X
Γ_R	Trajectories reachable by an adversary
Γ_{eff}	Effectively reachable trajectories, $\Gamma_{\text{eff}} \subseteq \Gamma_R$
Γ_{impact}	Trajectories terminating in an inadmissible state
$\text{Valid}(\cdot)$	Judgment that a certificate's claims remain supported by evidence E_t

Table 1: Notation used throughout the paper.

vocabulary, is the successful traversal of a trajectory into an inadmissible region, and defense is the elimination of such trajectories from the reachable set. What reachability analysis does not by itself provide is an account of evidence: a reachability proof establishes what a system's dynamics permit, not what a particular system's actual history was, or how that history can be verified after the fact by a party without access to the system's internals. This paper's contribution is not a new reachability method; it is the observation that security and trust, unlike classical safety verification, are evaluated retrospectively and by outside parties, which is why the certificate — an evidentiary object, not merely a dynamical one — becomes necessary.

Byzantine fault tolerance and distributed consensus. Distributed systems theory addresses a closely related problem: how a collection of agents, some of which may behave arbitrarily or maliciously, can agree on a consistent shared history despite that adversarial presence. The consensus literature contributes the idea, used informally in Section 8, that a ledger's validity is a property of its full reconstructible chain rather than of any single snapshot. What this paper adds is not a new consensus protocol, but a generalization of the reason such protocols work: they succeed because they make $\mathcal{A}(H_\Sigma)$ verifiable without full replay, which is exactly the role this paper assigns to certificates. Byzantine fault tolerance is, in this framework's terms, one highly developed engineering answer to the question of how one constructs C_Σ for a distributed ledger, rather than a competing theory of what admissibility is.

Chain of custody. Evidence law's doctrine of chain of custody is, in this paper's terms, the oldest and most literal instance of certifying a history rather than a state: an item of evidence is admissible not because of any property intrinsic to the item, but because its custodial history — who held it, when, and under what documented conditions — can be reconstructed without gaps. This paper generalizes the doctrine's structure beyond physical evidence and legal proceedings, but the doctrine itself already contains, in embryonic and undertheorized form, the central claim of Section 2: that a break in the recorded history, not any change in the object's present condition, is sufficient to render it inadmissible.

None of these three fields is subsumed or superseded by what follows. Each contributes a piece of machinery: reachability contributes the geometry of trajectories, Byzantine fault tolerance contributes mechanisms for verifiable agreement under adversarial conditions, and chain of custody contributes the evidentiary intuition that state alone cannot certify a history. What this paper adds is a single formal object, the admissibility certificate, and the observation, developed fully in Sections 4 through 7, that security, provenance, reconstruction, and repair — ordinarily treated as separate concerns requiring separate machinery — are special cases of maintaining $\mathcal{A}(H_\Sigma)$ over time, evidenced by C_Σ, and recursively self-consistent by the Proposition of Section 7.1.

4 An Empirical Case Study: Cox's Model Security Portfolio

The formal apparatus of Sections 1–3 was developed without reference to any particular domain. We now turn to a concrete industrial framework and ask a specific question: does anything resembling $\mathcal{A}(H_\Sigma)$ and its supporting evidence already exist in practice, independently of this paper's vocabulary?

Susanna Cox's *Securing AIML Systems in the Age of Information Warfare* [1] provides an unusually clean answer. Cox is not attempting to formalize admissibility, define system histories, or develop a general theory of trust. She is solving a narrower, practical problem: how should an organization secure machine learning systems operating in an adversarial information environment, using the tools already available to MLOps practice. That narrower ambition is precisely what makes the paper useful here. Its correspondence to the framework of Section 2 was not built in; if it exists, it was discovered independently, under the pressure of practical necessity rather than theoretical motivation. A correspondence arrived at twice, by two unrelated routes, is stronger evidence for the underlying structure than either route alone.

The correspondence is close to exhaustive.

Trusted baseline → admissible state. Cox's framework depends throughout on the existence of a documented "trusted state" for a model — a reference configuration against which later behavior is compared. This is admissibility in embryo: a trusted state is simply a state whose history is, at the time it is recorded, believed admissible. What Cox's framework lacks is the recognition that the baseline's authority derives entirely from the history behind it, not from any property of the state itself — which is exactly the blind spot the identical-endpoint problem of Section 2 exposes.

Monitoring thresholds → admissibility predicates. Cox's continuous-monitoring pipeline compares live inference behavior, data statistics, and performance metrics against baseline

values, triggering review when thresholds are crossed. This is an attempt to approximate $\mathcal{A}(H_\Sigma)$ using observable proxies, since the full history is not directly inspectable at runtime. The thresholds are not the admissibility condition itself; they are a computable surrogate for it, chosen because the true predicate cannot be evaluated directly on a live system.

FMEA → admissible-failure decomposition. Failure Modes and Effects Analysis, as Cox adapts it, is an attempt to enumerate in advance the ways a history can become inadmissible — the transitions, interventions, or data events that would move H_Σ out of $\mathcal{H}_{\text{adm}}$. It is, in effect, a partial specification of the constraint set defining admissibility for a given system, built before deployment and revised as new failure modes are discovered.

Datasheets and model cards → reconstruction metadata. These documents record provenance, composition, and intended use for data and models respectively. In the vocabulary of Section 2, they are fragments of L_t and D_t made explicit and durable — exactly the evidence required to reconstruct H_Σ after the fact, when a system's admissibility must be evaluated retrospectively rather than assumed [2, 3].

Lineage tracking → reachability information. Cox's emphasis on lineage analysis, particularly during retraining, is a direct attempt to answer the question Section 6 poses formally: from which prior states could the present state have been reached, and via which of those paths? Lineage records are the empirical trace of H_Σ itself, or as much of it as the organization chose to preserve.

Incident response → repair operator. When Cox's framework detects a threshold breach, it routes the system to human review and, potentially, to retraining or rollback. This is an attempt at what Section 5 distinguishes carefully as repair, though Cox's framework, like most of the literature it draws on, does not clearly distinguish repair from mere reconstruction, and this is one of its more consequential gaps.

Human review → admissibility judgment. Where automated thresholds are insufficient to resolve whether a given history remains admissible, Cox's framework escalates to a human reviewer. This is a tacit acknowledgment that $\mathcal{A}(H_\Sigma)$ is not, in general, mechanically decidable from the available evidence — that some judgments about historical admissibility require interpretation the automated system cannot supply.

Taken individually, each of these correspondences might be dismissed as coincidental — after all, any reasonably thorough security framework must address provenance, monitoring, and recovery in some form. Taken together, however, the correspondence is total: every major structural component of Cox's framework maps onto a distinct piece of the admissibility apparatus, and none of Cox's components lacks a counterpart in that apparatus. This is not the pattern one would expect from two frameworks addressing merely adjacent problems. It is the pattern one would expect from two independent attempts to operationalize the same underlying mathematical object, using different vocabularies and arriving from different directions — one from formal constraint theory, the other from the practical demands of securing production ML systems.

What Cox's framework does not do — and what the remainder of this paper supplies — is name the object it has built, distinguish sharply between the different grades of recovery it invokes under a single word ("recovery," "retraining," "rollback," used almost interchangeably), or recognize that the true target of every one of these mechanisms is not the model's present state but the admissibility of the history that produced it. Making that structure explicit is the task of Sections 5 through 7. We begin with the distinction Cox's

own framework needs most and has least: the difference between reconstructing a system and repairing one.

5 Reconstruction Versus Repair

Cox's framework, in common with most of the security and MLOps literature it draws on, uses "recovery," "retraining," and "rollback" in ways that shade into one another without clear boundaries. This is not a minor terminological looseness. It conceals a hierarchy of operations with sharply different guarantees, and collapsing that hierarchy is one of the more consequential ways a security framework can fail without appearing to.

Rollback. The narrowest operation is rollback: replacing the current parameters θ_t with a previously stored $\theta_{t'}$, $t' < t$, typically retrieved from a version-controlled artifact repository. Rollback restores an artifact. It says nothing about $D_{t'}$, $L_{t'}$, $G_{t'}$, or any other component of $\Sigma(t')$ — only that the parameter values match what was once recorded. Rollback is fast, mechanical, and requires no judgment about why the system needed restoring in the first place.

Reconstruction. A broader operation attempts to restore not merely $\theta_{t'}$ but the fuller state $\Sigma(t')$ — retraining the model against the data and pipeline configuration recorded as having produced $\theta_{t'}$, so that the resulting system is not merely artifact-identical but state-equivalent to some earlier point in the system's history. Reconstruction is what Cox's framework generally means by "retraining from a validated baseline." It is a substantially stronger guarantee than rollback: it re-derives the state rather than merely fetching a stored copy of it.

Repair. The broadest and most demanding operation in this hierarchy restores not a state but admissibility to the history. More precisely:

Definition 2 (Repair)**.** *Repair is a constrained transformation that achieves a required system objective while preserving those structures whose continuity defines the system's identity.*

In the security context that motivates this paper, the required objective is the closure of a compromise mechanism, and the preserved structures are the system's legitimate capabilities and prior commitments; repair restores forward-going admissibility by ensuring the mechanism that produced the departure has itself been closed. But nothing in the definition restricts it to correction. A system can equally undergo repair in order to acquire a capability it previously lacked, provided the transformation stays within whatever preserves its identity-defining structure — the objective achieved need not be a return to a prior condition at all. What distinguishes repair from mere reconstruction in either case is not the direction of the change but the presence of a preservation constraint alongside the objective: reconstruction re-derives a state; repair re-derives a state subject to a boundary on what must not move.

Reconstruction can faithfully reproduce $\Sigma(t')$ and still fail to repair the system if the conditions that made H_Σ inadmissible in the first place (a compromised upstream data

source, a still-active feedback loop, an unrevoked insider credential, an unpatched ingestion pipeline) remain in place. In that case, reconstruction simply returns the system to the mouth of the same trap. The history resumes exactly where it left off, and nothing prevents it from retracing the same inadmissible path forward. Repair requires, in addition to reconstructing a trusted prior state, that the mechanism by which H_Σ left $\mathcal{H}_{\text{adm}}$ has itself been identified and closed: the forward continuation of the history, not merely its restored starting point, must be admissible.

This gives a strict hierarchy of recovery operations.

Proposition 1 (Hierarchy of Recovery)**.** *Every repair operation contains a reconstruction, and every reconstruction that restores parameter values contains a rollback. Consequently,*

$$\textit{Rollback} \subsetneq \textit{Reconstruction} \subsetneq \textit{Repair}. \tag{5}$$

Every repair operation includes a reconstruction (some prior state must be restored or re-derived), and every reconstruction that restores parameter values includes a rollback (the parameters must, at minimum, match some earlier recorded configuration) — but neither containment reverses. A system can be rolled back without being reconstructed (only θ matches; the surrounding state does not). A system can be reconstructed without being repaired (the prior state is faithfully rebuilt; the vulnerability that compromised it is not addressed).

Why Cox's framework needs this distinction and does not have it. Cox's continuous-training pipeline routes retraining triggers through data validation and, when security criteria are met, human review — but the framework's account of what makes retraining safe to proceed with is comparatively thin. It validates the data being used for the new training run against baseline thresholds, which addresses one channel by which inadmissibility could re-enter the system, but it does not require, as a formal precondition of resuming training, that the original compromise mechanism has been diagnosed and closed. A poisoning attack that entered through a still-active data-scraping vulnerability, for instance, could in principle survive a full retraining-from-baseline cycle intact, since the vulnerability lies in P_t (the pipeline) rather than in $D_{t'}$ (the specific dataset being validated against a threshold). Cox's framework would, in this scenario, correctly reconstruct the system and incorrectly report it as recovered.

This is not a flaw specific to Cox; it is close to the default assumption of the security literature generally, where "recovery" is treated as synonymous with "restoration to a known-good state" and the adequacy of that restoration is judged by how faithfully the state was reproduced rather than by whether the forward trajectory from that state is now admissible. The distinction matters most precisely in the cases a security framework is built to catch: sophisticated, persistent, or environmentally-embedded attacks, where the compromise is not a single corrupted artifact but an ongoing condition of the system's surrounding history.

A consequence for certification. This hierarchy has a direct bearing on the admissibility certificate introduced formally in Section 7. A certificate that records only that a rollback or reconstruction occurred — "restored to baseline as of date t'" — certifies strictly less than

one that records the diagnosed mechanism of compromise and evidence that it has been closed. The former certifies a state. The latter certifies a history's continued admissibility going forward. Only the latter is repair in the sense this paper intends, and only the latter is sufficient grounds for resuming trust.

With this hierarchy established, we can now return to the attack side of the problem and ask what it means, formally, for an adversary to move a system's history out of $\mathcal{H}_{\text{adm}}$ in the first place — and for a defender to prevent that path from existing. This is the reachability question of Section 6.

6 Reachability

Cox's own formalization of the attacker's problem is Boolean: a successful attack requires access, knowledge, capability, and impact jointly, with each top-level requirement satisfiable through any of several alternative lower-level routes, connected by disjunction. This is a genuinely useful decomposition for organizing a threat model, and nothing in this section is meant to discard it. But it describes a checklist, not a space. It tells a defender which categories of precondition an attacker must satisfy; it does not tell them, geometrically, what an attack actually is, or what it would mean to close one off with confidence rather than merely make it harder.

The framework of Section 2 supplies that geometry directly, because we have already established that the relevant object is not a state but a trajectory.

Attack trajectories. Let X denote the space of possible system histories, and let

$$\Gamma = \{\gamma : [0, 1] \to X\} \tag{6}$$

denote the set of all trajectories through that space. Let $\Gamma_R \subseteq \Gamma$ denote the subset reachable by some sequence of actions available to an adversary, given the access, knowledge, and capability constraints that apply to them.

An attack succeeds, in this formalization, iff

$$\exists\, \gamma \in \Gamma_R \text{ such that } \gamma(0) \in X_0 \text{ and } \gamma(1) \in X_{\text{impact}}, \tag{7}$$

where X_0 is the system's current (present, ostensibly admissible) history-state and X_{impact} is the set of history-states an attacker would regard as a successful outcome.

Cox's four Boolean preconditions can now be read as constraints on which trajectories belong to Γ_R at all, rather than as independent gates a checklist must tick. Access constraints determine which transitions γ is permitted to make. Knowledge constraints determine which transitions can be chosen deliberately rather than stumbled into. Capability constraints bound the rate and complexity of transitions available. Impact is simply the requirement that the endpoint lie in X_{impact}. The Boolean AND across categories, and OR within them, falls out automatically: a trajectory is only in Γ_R if it simultaneously respects all four constraint types, and each constraint type may be satisfiable through multiple distinct sub-routes.

Nominal versus effective reachability. Not every trajectory in Γ_R is, in practice, equally available to Red. An adversary's *nominal* freedom of action, everything access, knowledge, and capability constraints do not explicitly forbid, is broader than their *effective* freedom: the subset of Γ_R that remains viable once the trajectory is also required to avoid triggering detection, preserve the attacker's own operational continuity, or hold together under whatever other constraints the adversary is themselves subject to. Write

$$\Gamma_{\text{eff}} \subseteq \Gamma_R \tag{8}$$

for this effectively reachable subset. Red's actual problem is not $\exists\gamma \in \Gamma_R$ but $\exists\gamma \in \Gamma_{\text{eff}}$ connecting X_0 to X_{impact} — and Blue's task correspondingly is not to eliminate $\Gamma_R \cap \Gamma_{\text{impact}}$ in its entirety, which may be infeasible, but to reduce or disconnect $\Gamma_{\text{eff}} \cap \Gamma_{\text{impact}}$, a substantially smaller and more tractable target. This same distinction explains a fact that would otherwise be puzzling on the defender's side: many nominally available repairs — transformations that would technically satisfy Section 5's admissibility criterion — are not operationally available either, because they violate some preservation constraint the system cannot afford to relax. Nominal and effective reachability diverge for attacker and defender alike, for the same structural reason. It is worth being explicit that Γ_{eff} is not a static subset carved out by resources or capability alone; it depends on the history traversed so far, since what an actor can do next without triggering detection or violating their own preservation constraints is itself a function of what they have already done. Effective reachability is history-dependent in exactly the sense Section 2 gives that term, not merely a smaller capability-bounded region of Γ_R fixed once and for all.

Defense as trajectory elimination. This reformulation changes what it means to defend. On the checklist reading, Blue's task is to make each of four boxes harder to check. On the trajectory reading, Blue's task is to act on Γ_{eff} itself: Blue succeeds iff $\Gamma_{\text{eff}} \cap \Gamma_{\text{impact}} = \varnothing$. Blue does not need to eliminate every trajectory in Γ_R — only enough of the effectively reachable ones that none remaining reaches X_{impact}. This makes explicit that hardening one access route while leaving a disjunctive alternative open (exactly the OR-structure Cox herself identifies within each category) accomplishes nothing if $\Gamma_{\text{eff}} \cap \Gamma_{\text{impact}}$ remains nonempty through the unclosed route.

Why this matters for admissibility. The connection to Sections 2–5 is direct. X_{impact}, in the vocabulary already established, is simply the set of histories H_Σ that fail $\mathcal{A}(H_\Sigma)$: inadmissible histories reached via a trajectory an attacker deliberately steered toward. An attack, on this account, is not fundamentally a technical event (a poisoned batch, an exfiltrated credential) so much as the successful traversal of a trajectory from an admissible history into an inadmissible one, using whichever access, knowledge, and capability the attacker could assemble. This is why Section 5's hierarchy matters here too: rollback and reconstruction attempt to move the system back toward X_0, but if the trajectory that reaches X_0 from the current position still lies within Γ_{eff} — if the route the attacker used remains effectively open — nothing prevents the same or a similar trajectory from being retraced. Repair, in the sense of Section 5, is precisely the operation of closing the relevant portion of Γ_{eff}, not merely relocating the system to a point it once occupied.

A timing refinement. One further constraint deserves brief mention: Blue's elimination of $\Gamma_{\text{eff}} \cap \Gamma_{\text{impact}}$ must, in an adaptive system, occur faster than an attacker can traverse a trajectory to X_{impact} and exploit the arrival before detection. This is the content of Cox's OODA-loop discussion, and it can be stated as a response-time inequality,

$$\tau_{\text{detect}} + \tau_{\text{interpret}} + \tau_{\text{decide}} + \tau_{\text{act}} < \tau_{\text{attack}}, \tag{9}$$

with a stronger, damage-weighted version bounding accumulated harm along γ before response completes rather than merely elapsed time. We do not develop this direction further here, since it is orthogonal to the paper's central claim; it is noted because it explains why Γ_{eff} must be understood as changing over time, not as a fixed set computed once at design time and never revisited.

With the attacker's problem now stated geometrically — as the search for a trajectory into inadmissibility — and the defender's problem correspondingly restated as trajectory elimination, we are in a position to define formally the object that supports both: a record sufficient to establish, for a given history, whether it lies in $\mathcal{H}_{\text{adm}}$, and sufficient to support the trajectory-elimination judgments this section has described. That object is the admissibility certificate.

7 Admissibility Certificates

Section 2 established that admissibility must be evaluated over a system's history H_Σ rather than its instantaneous state. This raises an immediate practical difficulty: H_Σ is, in general, neither fully observable nor exhaustively recorded, and $\mathcal{A}(H_\Sigma)$ cannot be evaluated directly by inspecting the past in full. We now introduce the object that resolves this difficulty in practice.

The certificate as abstract object. For a system Σ, define the admissibility certificate C_Σ as a structured, versioned body of evidence sufficient to support a judgment about the admissibility of H_Σ, without requiring H_Σ to be replayed or re-observed in full. At this level, C_Σ is deliberately left abstract: nothing about its internal structure has yet been fixed, only its role — it stands as evidence for $\mathcal{A}(H_\Sigma)$, in the way a proof stands as evidence for a theorem without being identical to the fact it establishes.

One concrete realization. A certificate must, in practice, take some representable form. One useful and sufficiently general realization (the one this paper develops, though not the only one a given domain might adopt) decomposes C_Σ into seven components:

$$C_\Sigma = (\mathcal{P}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{B}, \mathcal{Q}, \mathcal{R}), \tag{10}$$

where $\mathcal{P}$ is provenance, $\mathcal{L}$ is lineage, $\mathcal{F}$ is the anticipated-failure decomposition, $\mathcal{E}$ is empirical adversarial evidence, $\mathcal{B}$ is the trusted baseline set, $\mathcal{Q}$ is the set of monitoring and escalation predicates, and $\mathcal{R}$ is the graded recovery record, distinguishing rollback, reconstruction, and repair per Section 5. We refer to this as the *minimal operational decomposition* rather than as a canonical or exhaustive one. A domain with additional structure — an eighth field

capturing, say, regulatory sign-off, or a domain-specific notion of witnessed observation — extends this decomposition rather than contradicting it; the theory's content lies in the existence and role of C_Σ, not in the count of its coordinates.

What the certificate certifies. It is worth being precise about what C_Σ is evidence for. The certificate does not certify Σ_t, the system's present state. It certifies H_Σ, the history that produced that state. This is the notational commitment Section 2 requires: a certificate that referred only to Σ_t would silently reintroduce the state-only framing this paper exists to reject. Two systems with $\Sigma_1(T) = \Sigma_2(T)$ can, and in the adversarial case typically will, carry certificates $C_{\Sigma_1} \neq C_{\Sigma_2}$ — and it is precisely this difference, invisible to any state-only inspection, that the certificate exists to make legible.

Admissibility as a maintained relation. A certificate is not issued once and trusted indefinitely. We define admissibility at time t as

$$\Sigma_t \in \mathcal{A}_t \iff \text{Valid}(C_{\Sigma_t}, E_t) = 1, \tag{11}$$

where E_t is the evidence available at time t and $\text{Valid}(\cdot)$ is the judgment — automated where $\mathcal{Q}$ permits, human where it does not, per Section 4's account of Cox's escalation mechanism — that the certificate's claims remain supported by current evidence. Admissibility is therefore a relation that must be continuously re-established, not a property conferred once at deployment. This is the formal counterpart of Cox's continuous-monitoring architecture: monitoring exists because $\text{Valid}(\cdot)$ must be re-run, not run once.

Cox's portfolio as one implementation. Every component of Cox's model security portfolio, mapped in Section 4, is now identifiable as a particular implementation choice for one coordinate of the minimal operational decomposition: datasheets and model cards implement $\mathcal{P}$; lineage analysis implements $\mathcal{L}$; FMEA output implements $\mathcal{F}$; adversarial testing results implement $\mathcal{E}$; documented trusted states implement $\mathcal{B}$; monitoring thresholds implement $\mathcal{Q}$; and incident-response records implement $\mathcal{R}$ — though, per Section 5, incompletely, since Cox's $\mathcal{R}$ does not cleanly separate rollback, reconstruction, and repair. This is the sense in which Cox's framework is evidence rather than inspiration: coordinate by coordinate, her portfolio already is an instance of C_Σ, arrived at without the general theory that would have told her so.

7.1 The Regress Problem and Its Resolution

A natural objection presents itself as soon as C_Σ is introduced: if trust in a system requires a certificate, what warrants trust in the certificate itself? An attacker unable to compromise Σ directly might instead forge C_Σ — fabricate provenance, backdate a baseline, manufacture adversarial-testing evidence — and nothing said so far explains why this should be any harder than compromising Σ was in the first place. Pushed further, the objection threatens to become fatal: if certificates require certification, and that certification requires certification in turn, admissibility appears to rest on an infinite regress with no evident stopping point.

The apparent regress is resolved by the history formalism introduced in Section 2, and requires no new machinery — only the observation that C_Σ was never exempt from the framework's own definitions. A certificate is a system artifact like any other: it is produced by a process, that process has a history, and that history is exactly the kind of object Section 2 already knows how to evaluate. Write $H(C_\Sigma)$ for the certificate's own history — the record of how C_Σ was compiled, by what process, from what underlying evidence, under whose authority, with what audit trail. Admissibility extends to the certificate exactly as it applies to any other history: $\mathcal{A}(H(C_\Sigma))$.

This is not a new predicate invented to patch the objection. It is $\mathcal{A}(\cdot)$ from Section 2, applied one level up. Nothing in $\mathcal{A}$'s original definition restricted it to histories of "primary" systems as opposed to histories of the artifacts used to evaluate them; that restriction existed only in how the framework was first presented, not in its content.

$$\mathcal{A}\big(H(C_\Sigma)\big) \tag{12}$$

The full progression, stated once and explicitly: a system Σ has a history H_Σ; a certificate C_Σ is attached to that system as evidence of $\mathcal{A}(H_\Sigma)$; the certificate itself has a history $H(C_\Sigma)$; and the same predicate $\mathcal{A}$ applies to that history in turn. Nothing beyond the machinery already introduced in Section 2 is required.

Proposition 2 (Closure under Certification)**.** *If certificates are system artifacts, then admissibility extends to certificate histories by the same predicate $\mathcal{A}$. Consequently, admissibility induces a finite directed acyclic evidentiary graph over evidentiary artifacts, rather than an infinite chain.*

The finiteness and acyclicity claims rest on nothing more than the observation that certificate-issuing processes are, as a matter of practice, finite and non-self-referential: at some point a chain of "what certifies this evidence" terminates not in a further certificate but in an object the framework does not attempt to certify further, such as a cryptographic signature verified against a root key, a measurement taken by a physical instrument, an attestation made by an institution the domain has agreed to trust, a fact established by direct witnessed observation, or a legal instrument accepted as authoritative within its jurisdiction. These are the graph's leaves: trust anchors external to the framework, present in every evidentiary system — cryptographic, legal, scientific, institutional — whether or not that system names them explicitly. Edges in this graph represent evidentiary dependence, and no certificate depends, even indirectly, on itself; a certification scheme that permitted such a cycle would not be circular reasoning in the abstract but a concrete forgery vector, since a cycle could be satisfied by mutual assertion alone, with no trust anchor required anywhere in the loop.

The force of this result is not that it eliminates the need for trusted assumptions; no framework can do that. It is that the assumptions become visible and locatable rather than smuggled in unnamed. A security architecture that declares a model trustworthy "because it has a certificate," without specifying $H(C_\Sigma)$'s own trust anchors, has not escaped the regress: it has merely stopped looking one level too early. A certificate whose own history is undocumented is, by the framework's own standard, no more admissible than the system it purports to certify, since a certificate is only as strong as the shortest undocumented edge in its dependency graph. This also clarifies what was left implicit earlier in this section:

the coordinates of C_Σ — $\mathcal{P}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{B}, \mathcal{Q}, \mathcal{R}$ under the minimal operational decomposition — are not independently self-evident; each is itself an artifact with a compilation history, and it is the Proposition above that licenses treating them as evidence at all, rather than as unexamined assertion. A datasheet is admissible evidence of $\mathcal{P}$ only insofar as the process that produced the datasheet is itself auditable — which is, again, the same predicate, one level up.

8 Generalization

We can now state the paper's central claim — the theorem toward which Sections 1 through 7 have been building, and which the remainder of this section exists to illustrate rather than further argue for.

Principle 1 (History Principle)**.** *Any adaptive system whose future evolution depends on accumulated information requires admissibility to be defined over histories rather than instantaneous states.*

The proof, such as it is, is not a formal derivation but an observation already established piecewise across Sections 2 through 7: whenever a system's future behavior is a function of its past — whenever $\Sigma(t+\delta)$ depends not just on $\Sigma(t)$ but on how $\Sigma(t)$ was produced, because that production process left traces (data dependencies, learned parameters, structural commitments, accumulated damage or repair) that continue to influence what the system can validly become — state-only admissibility checks become blind to exactly the failures that matter most. The identical-endpoint problem of Section 2 is not a special pathology; it is the generic situation whenever $\Sigma(t)$ is a compressed function of a much richer H_Σ, which is to say, whenever the system has learned, adapted, accreted, or been maintained rather than simply existing as a fixed given.

What follows is not an exhaustive survey but a demonstration that the same structure recurs, largely unmodified, across domains that do not ordinarily share a technical vocabulary.

Operating systems and software. A running process's memory state can be bit-identical whether it arrived there through a validated code path or a successful exploit. Thompson's classic account of a compiler that inserts a backdoor into any program it compiles — including, self-perpetuatingly, into future copies of itself — is exactly this problem: a compromised and an honest toolchain can produce bit-for-bit identical binaries, so no inspection of the binary alone certifies how it was built [4]. Reproducible-build efforts respond directly, treating an auditable compilation history as evidence that a binary corresponds to its claimed source, rather than trusting the binary as a terminal object [5] — provenance-based verification certifying H_Σ rather than $\Sigma(t)$.

Aviation and safety-critical infrastructure. An aircraft judged airworthy is not judged so on its current physical configuration alone; airworthiness is constituted by a maintained history. Under United States federal aviation regulation, owners and operators must retain records of total time in service, the current status of applicable airworthiness directives with

method and date of compliance, and documentation of major alterations — and an aircraft may not be certificated or used if these records are lost without being reconstructed [8]. Two airframes in visually identical condition can differ entirely in whether that condition was reached through admissible or falsified service. The paperwork is not overhead; in this framework, it is the certificate, and its absence is equivalent to inadmissibility regardless of how sound the aircraft appears.

Biological evolution. A phenotype's robustness at time t depends not only on its present configuration but on the developmental trajectory that produced it. Waddington's concept of canalization describes development buffered so a consistent adult phenotype emerges despite perturbation [6] — two organisms can reach an outwardly identical phenotype through histories differing in how thoroughly perturbations were absorbed, with consequences for future robustness the endpoint alone does not reveal.

Scientific reproducibility. A published result is not admissible on the reported finding alone; it is admissible only insofar as the methodological history producing it — data collection, analysis choices, exclusions — is available for scrutiny. A large-scale attempt to replicate one hundred psychology studies found replication effect sizes averaging roughly half the magnitude of the originals, despite the originals overwhelmingly reporting significant results [7]. Two studies reporting identical results can differ completely in admissibility depending on whether their methodological history was honestly recorded, which is why the reproducibility crisis is a crisis of unrecoverable histories rather than of false endpoints.

Governance and legal chains of custody. Evidence is inadmissible in the technical legal sense not because of any property of the evidence itself but because of a break in its custodial history — the doctrine's own name literally anticipates H_Σ: an unbroken, documented trajectory from origin to present, without which the object's current state is not permitted to certify anything.

Distributed consensus. A blockchain's validity is defined not by any single ledger snapshot but by the reconstructibility of the entire chain back to genesis under the protocol's rules; Nakamoto's design secures a distributed ledger through a timestamp server and proof-of-work, producing a tamper-resistant record rather than relying on trust in any single copy [9]. A forged snapshot identical to the true present state is worthless precisely because consensus protocols continuously re-validate $\mathcal{A}(H_\Sigma)$ rather than $\Sigma(t)$.

Civilizational continuity. At the largest scale, institutions, legal systems, and bodies of technical knowledge persist as admissible only insofar as their present form remains reconstructible from a documented, transmissible history — a civilization that has lost the ability to explain how its present institutions came to be, or why particular practices were adopted, has not necessarily changed its present configuration, but has lost the evidentiary basis for judging that configuration's continued admissibility, which is a different and in some ways more serious form of loss.

None of these domains borrowed the vocabulary of this paper, and none of them, so far as we are aware, has been described using a shared formal object. What they share is the structural fact the History Principle names: wherever a system's trustworthiness cannot be read off its present state alone, some version of H_Σ, $\mathcal{A}(H_\Sigma)$, and C_Σ is already doing quiet, usually unnamed work — as maintenance logs, chains of custody, reproducibility standards, or consensus protocols. Cox's AI-security framework, examined in Sections 4 through 7, is simply the instance closest to this paper's original motivation. It is not the phenomenon's natural home. It is one room in a much larger house.

Every domain surveyed above shares one further feature worth isolating before closing this section: in each case, admissibility was being verified for a history that already existed, using evidence external to the system under evaluation — an inspector, an auditor, a replication team, a verifying node. This raises a question the History Principle does not by itself answer. If having an admissible history is what distinguishes a genuine derivation from a fluent imitation of one, is this only detectable by an outside verifier with independent evidence, or can a system's own output reveal, from the inside, whether it has one? The remainder of this section develops that question directly, using the same apparatus in a different direction: not certifying a claimed history against outside evidence, but testing whether an agent's behavior is consistent with having one at all.

8.1 Reflexive Application: Reconstruction as a Selection Signal

The apparatus developed so far treats admissibility as a property to be verified externally: a certificate accompanies a system, and a verifier asks whether the history it certifies remains trustworthy, using evidence largely independent of the system itself. There is a second, reflexive use of the same apparatus, motivated directly by the identical-endpoint problem of Section 2. That problem showed that a system's endpoint alone cannot certify its history to an outside observer; the present subsection asks a sharper version of the same question about a system that is itself supposed to have traversed an admissible history — a trained model — and shows that the identical-endpoint problem can be deliberately manufactured as a diagnostic, revealing whether the model's own behavior is consistent with having represented that history at all, rather than merely having learned to produce its endpoint.

Withheld-history reconstruction. Consider a document with a genuine revision history $H_E = \{E(t) : t \in I\}$ — successive drafts of a technical essay, in which arguments, derivations, or intermediate results were introduced in a particular order, some steps depending on others having already been established. The final draft $E(T)$ is observable. Now construct a task in which part of H_E is withheld from a model: the ordering of intermediate steps, the presence of a step at all, or the dependency structure connecting them. The model is asked to reconstruct the missing portion of the history from $E(T)$ and whatever partial evidence remains.

This is a deliberately manufactured instance of the identical-endpoint problem from Section 2. In general there are many candidate histories that could plausibly terminate at something resembling $E(T)$ — many orderings of steps that would read as fluent, many omissions that would go unnoticed by a reader checking only the final draft — and only one of them is admissible, in the sense of actually reflecting the derivational dependencies

the argument requires. A step that logically depends on an earlier result cannot, in an admissible history, precede that result, regardless of how natural the reordering reads in isolation.

Why reconstruction errors are diagnostic rather than incidental. A model that recovers the wrong order, or silently supplies a step that was not in fact necessary while omitting one that was, has not merely erred at the level of surface content. It has produced a different history that happens to terminate near the observable endpoint but does not respect the dependency structure that made the true history admissible. This is exactly the reconstruction-without-repair failure of Section 5, relocated from system security to model evaluation: the reconstruction can be fluent, plausible, and superficially indistinguishable from the true derivation, while failing the stronger requirement of tracking which transitions were actually load-bearing.

A selection criterion. This gives withheld-history reconstruction a use beyond evaluation: it can serve as a selection signal during training or model comparison. A model that reliably recovers the true ordering — and in particular, one that correctly identifies which steps are necessary to reach a given later result, rather than merely locally plausible continuations of the text — is exhibiting evidence of having represented the admissible derivational path, not just the surface form of its endpoint. A model whose reconstructions are fluent but structurally arbitrary can be discarded or downweighted on exactly this basis, using no signal beyond the withheld-history task itself.

This distinction — between knowing an endpoint and having internalized the admissible path to it — is not confined to essay drafts. It is a general test applicable to any domain in which correct final outputs can, in principle, be produced through more than one route, only some of which respect the actual dependency structure of the problem.

Framed against Section 7, the withheld-history task is a way of manufacturing $\mathcal{E}$ — empirical adversarial evidence, in the vocabulary of the minimal operational decomposition — for a claim that would otherwise have no certificate at all: the claim that a given model's competence rests on an admissible history of derivation rather than on memorized or pattern-matched endpoints. A model has no C_Σ in the ordinary sense; nothing external documents the history by which its outputs were derived at inference time. Withheld-history reconstruction is a way of generating exactly that missing evidence directly from behavior, in the same spirit as Cox's adversarial testing generates $\mathcal{E}$ for a deployed system by deliberately probing it rather than waiting for an incident to occur.

What the reflexive case makes visible is that the distinction this paper has been drawing — between a state and the history that produced it — is not a peculiarity of AI security, biological systems, or infrastructure. It is a condition on any process in which correct outputs underdetermine the path that generated them, and in which some of the discarded paths are, in a precise sense, better than others: more efficient, more robust, more faithful to the structure of the underlying problem. Whether the object in question is a deployed model's training history, a bridge's maintenance record, or an essay's sequence of drafts, the same fact holds — the endpoint alone cannot certify the process, and evaluating only the endpoint discards exactly the information needed to tell a genuine derivation from a fluent imitation of one.

9 Conclusion

We began with a narrow observation: that a deployed model's present weights can satisfy every available check while having been produced by a compromised or undocumented process, and that state-only security frameworks are structurally unable to see this. Formalizing that observation required introducing system histories H_Σ as the primary object of analysis, admissibility $\mathcal{A}(H_\Sigma)$ as a predicate over histories rather than states, and admissibility certificates C_Σ as the evidentiary object by which that predicate can be approximated and continuously re-validated in practice.

Cox's *Securing AIML Systems in the Age of Information Warfare* served throughout as empirical confirmation rather than motivation: an industrial framework, built under no obligation to this paper's theory, that independently constructed nearly every coordinate of C_Σ under its own vocabulary. Reading it through this lens exposed a distinction the security literature generally lacks — the strict hierarchy of rollback, reconstruction, and repair — and a reachability reformulation of the attacker's problem that replaces a Boolean checklist with a geometry of trajectories through history-space, sharpened by the distinction between nominal and effective reachability.

Admissibility certificates do not replace cryptographic trust, institutional authority, provenance systems, or engineering practice. Rather, they provide a common mathematical language in which these mechanisms become instances of a more general theory of maintaining adaptive systems within admissible histories. AI security is not this paper's subject. It is the occasion on which the subject became visible.

References

[1] Cox, S. (2022). *Securing AIML Systems in the Age of Information Warfare*. Critical Alliance. https://doi.org/10.5281/zenodo.13905972

[2] Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2021). Datasheets for Datasets. *Communications of the ACM*, 64(12), 86–92.

[3] Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (FAT* '19), 220–229.

[4] Thompson, K. (1984). Reflections on Trusting Trust. *Communications of the ACM*, 27(8), 761–763.

[5] Lamb, C., & Zacchiroli, S. (2022). Reproducible Builds: Increasing the Integrity of Software Supply Chains. *IEEE Software*, 39(2), 62–70.

[6] Waddington, C. H. (1942). Canalization of Development and the Inheritance of Acquired Characters. *Nature*, 150, 563–565.

[7] Open Science Collaboration. (2015). Estimating the Reproducibility of Psychological Science. *Science*, 349(6251), aac4716.

[8] United States Federal Aviation Administration. *14 CFR §91.417 — Maintenance Records*. Code of Federal Regulations.

[9] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.