A MANIFESTO

Computational Trust

計算信任宣言

We believe that autonomous systems deserve the same standard of verifiable trust that human decisions have always required.

v2 · September 2026

"Trust, but verify."

— A principle for an era when humans made the decisions.

When machines make the decisions,

verification is all we have.

「信任，但驗證。」

——屬於人類做決策年代的原則。

當機器開始做決策，

驗證就是我們僅有的一切。

The Problem

問題

Something fundamental has changed in how the world computes.

For fifty years, computers executed instructions written by humans, supervised by humans, and accountable to humans. If a decision was questioned, a human could explain it. If a process was challenged, a human could defend it. The chain of accountability was clear: machine does, human answers.

That chain is breaking.

Today, autonomous systems make decisions that no human supervised, no human can fully explain, and no human may even be aware of until after the consequences unfold. AI agents select tools, route tasks, allocate resources, triage patients, execute trades, and make recommendations — billions of times per day, at a scale no human oversight can match.

These systems are not malicious. They are not broken. They are working as designed. But their decisions involve non-deterministic processes — randomness, sampling, probabilistic reasoning — and the processes that produced those decisions are invisible. They are not recorded. They are not verifiable. They are not provable.

This is not primarily a security problem — even a perfectly secured system produces decisions no outsider can verify. This is not a performance problem. The systems are fast and accurate. This is a **trust problem** — the inability to independently verify, after the fact, that a computational process ran as declared, from recorded inputs, without undisclosed alteration.

We call this problem **Computational Trust**.

> 某些根本性的事情已經改變了世界計算的方式。

五十年來，電腦執行由人類撰寫、由人類監督、對人類負責的指令。如果一個決策受到質疑，人類可以解釋它。如果一個過程受到挑戰，人類可以為其辯護。問責鏈是清晰的：機器執行，人類回答。

這條鏈正在斷裂。

今天，自主系統做出沒有人類監督、沒有人類能完全解釋、甚至可能在後果展開之前沒有人類意識到的決策。AI 代理選擇工具、路由任務、分配資源、分診患者、執行交易、做出推薦——每天數十億次，以任何人類監督都無法匹配的規模。

這些系統不是惡意的。它們沒有壞掉。它們按設計運作。但它們的決策涉及非確定性過程——隨機性、抽樣、機率推理——而產生那些決策的過程是不可見的。它們未被記錄。它們不可驗證。它們不可證明。

這主要不是安全問題——即使一個防護完美的系統，其產出的決策也沒有任何外部者能驗證。這不是性能問題。系統快速且準確。這是一個**信任問題**——無法在事後獨立驗證計算過程是否按宣告執行、輸入有據、未經未揭露之竄改。

我們稱這個問題為**計算信任**。

Five Beliefs

五個信念

We hold these beliefs to be foundational. They are not technical claims. They are statements about what we believe the world needs — and what we commit to building.

> 我們認為這些信念是根本性的。它們不是技術主張。它們是關於我們相信世界需要什麼——以及我們承諾建構什麼的聲明。

Trust must be proven, not promised.

信任必須被證明，而非被承諾。

When a system says "this decision was fair," that is a promise. When a system produces a cryptographic proof that any independent party can verify — without trusting the system, its operator, or any intermediary — that is trust. We believe that as machines gain authority over consequential decisions, promises are no longer sufficient. Proof must become the default.

> 當一個系統說「這個決策是公平的」，那是一個承諾。當一個系統產出密碼學證明，任何獨立方都可以驗證——無需信任系統、其操作者或任何中介——那才是信任。我們相信，隨著機器對重大決策獲得決策權，承諾不再足夠。證明必須成為預設。

Verification must be independent of the operator.

驗證必須獨立於操作者。

An audit log written by the system being audited is not verification — it is a second promise. True verification requires that any third party, without cooperation from the operator, can examine the evidence and confirm that the process ran as declared — same inputs, same steps, same order, same keys. If the operator must be trusted for the verification to work, it is not verification. It is faith.

> 由被稽核系統撰寫的稽核日誌不是驗證——它是第二個承諾。真正的驗證要求任何第三方，在沒有操作者配合的情況下，能夠檢查證據並確認過程確如宣告般執行——同樣的輸入、同樣的步驟、同樣的順序、同樣的金鑰。如果驗證要能運作就必須信任操作者，那它不是驗證。那是信仰。

Trust must be measurable, not binary.

信任必須是可量測的，而非二元的。

A random number from a single pseudo-random generator and a random number from three independent entropy sources with cryptographic verification and blockchain anchoring are not equally trustworthy. They should not be treated the same. Trust must carry a score — a quantitative, reproducible measure of quality — so that consumers, regulators, and auditors can make informed decisions about whether a given level of trust is appropriate for a given use case.

> 來自單一偽隨機生成器的隨機數與來自三個獨立熵源、經密碼學驗證和區塊鏈錨定的隨機數，並非同等可信。它們不應該被同等對待。信任必須附帶分數——品質的量化、可重現量測——使消費者、監管機構和稽核員能夠做出明智的決定：特定層級的信任是否適合特定的使用案例。

IV

Trust must be portable, not platform-locked.

信任必須可攜帶，而非鎖定於平台。

A trust assertion that can only be verified on one chain, one cloud, or one vendor's infrastructure is not trust — it is lock-in. Computational Trust must produce objects that are verifiable anywhere: on any cloud, any chain, any device, any jurisdiction — offline, air-gapped, years later. Trust that requires a specific provider to verify is a dependency. Trust that anyone can verify is infrastructure.

> 只能在一條鏈、一個雲或一個供應商的基礎設施上驗證的信任斷言不是信任——而是鎖定。計算信任必須產出在任何地方都可驗證的物件：任何雲、任何鏈、任何設備、任何司法管轄區——離線、氣隙隔離、多年之後。需要特定供應商才能驗證的信任是依賴。任何人都能驗證的信任才是基礎設施。

V

Trust must be open, not proprietary.

信任必須是開放的，而非專有的。

TLS did not become the foundation of internet security by being proprietary. OAuth did not become the standard for delegated authorization by being closed. JWT did not become the lingua franca of identity by being vendor-specific. If Computational Trust is to become a foundational layer of autonomous computing, it must be built as an open standard — specified, documented, and implementable by anyone. We believe that trust infrastructure, like communication infrastructure, must be a public good.

> TLS 不是靠專有而成為網際網路安全的基礎。OAuth 不是靠封閉而成為委託授權的標準。JWT 不是靠供應商特定而成為身份的通用語言。如果計算信任要成為自主計算的基礎層，它必須作為開放標準來建構——被規定、被記錄、任何人都可以實作。我們相信，信任基礎設施如同通訊基礎設施，必須是公共財。

Ten Principles

十項原則

The five beliefs define what we value. The ten principles define how we build. Any system that claims to provide Computational Trust should be evaluated against these principles.

> 五個信念定義了我們重視什麼。十項原則定義了我們如何建構。任何聲稱提供計算信任的系統都應根據這些原則進行評估。

1. Proof over assertion

證明優於聲明

Every claim of trustworthiness must be accompanied by independently verifiable cryptographic proof. Statements without proof are opinions, not trust.

每個可信度的聲明都必須附帶可獨立驗證的密碼學證明。沒有證明的聲明是意見，不是信任。

2. Provenance over obscurity

出處優於模糊

The origin of every input to a trusted computation must be traceable, recorded, and independently auditable. Hidden sources are not trusted sources.

受信任計算的每個輸入的來源都必須是可追溯、有記錄和可獨立稽核的。隱藏的來源不是受信任的來源。

3. Scoring over pass/fail

評分優於通過/失敗

Trust quality must be expressed as a continuous, multi-dimensional score — not a binary gate. Different use cases require different trust levels. The score must be deterministically reproducible from the evidence.

信任品質必須以連續的、多維度的分數表達——而非二元閘門。不同使用案例需要不同的信任層級。分數必須從證據中確定性地可重現。

4. Objects over logs

物件優於日誌

Trust evidence must be packaged as self-describing, portable, verifiable digital objects — not appended to operator-controlled log files. A trust object carries its own proof. A log entry carries the writer's assertion.

信任證據必須被封裝為自描述、可攜帶、可驗證的數位物件——而非附加到操作者控制的日誌檔案。信任物件攜帶自己的證明。日誌條目攜帶撰寫者的聲明。

5. Offline over callback

離線優於回呼

Verification must be possible without contacting the issuer, the network, or any third party. A trust object that requires a phone-home to verify is a service dependency, not infrastructure. Trust must work air-gapped. Offline verification covers everything the artifact carries; anchor confirmation against a live chain is an additional assurance tier, not a prerequisite.

驗證必須在不聯繫發行者、網路或任何第三方的情況下可行。需要回撥才能驗證的信任物件是服務依賴，而非基礎設施。信任必須在氣隙隔離環境下運作。離線驗證涵蓋成品自身攜帶的一切；對照即時鏈的錨定確認是額外的保證層級，而非前提。

6. Composition over isolation

組合優於隔離

Trust must be composable. A trust object from one computation must be usable as input to another, creating verifiable trust chains. Complex systems are built from simple, individually verifiable units. Trust that cannot compose cannot scale.

信任必須可組合。一個計算的信任物件必須可用作另一個計算的輸入，建立可驗證的信任鏈。複雜系統由簡單的、可個別驗證的單元構建。無法組合的信任無法擴展。

7. Degradation over failure

降級優於失敗

When trust conditions cannot be fully met — a source is unavailable, a score falls below threshold, an anchor fails — the system must degrade transparently, not fail silently. Degradation governs evidence generation: reduced trust must be visible in the artifact, never hidden. Verification verdicts are the one exception — a verifier that cannot complete its checks must refuse, not guess. Honest degradation is trustworthy. A confident wrong answer never is.

當信任條件無法完全滿足——來源不可用、分數低於閾值、錨定失敗——系統必須透明地降級，而非靜默地失敗。降級適用於證據的產生：降低的信任必須寫在成品裡，絕不隱藏。驗證裁決是唯一的例外——無法完成檢查的驗證器必須拒絕，而非猜測。誠實的降級是可信賴的。自信的錯誤答案永遠不是。

8. Standards over products

標準優於產品

The trust object format, scoring methodology, signing protocol, and verification logic must be open, documented, and implementable by anyone. A single vendor's proprietary trust format is a product. An open, interoperable trust format is infrastructure. We build infrastructure.

信任物件格式、評分方法、簽名協議和驗證邏輯必須是開放的、有文件記錄的、任何人都可以實作的。單一供應商的專有信任格式是產品。開放的、可互操作的信任格式是基礎設施。我們建構基礎設施。

9. Lifecycle over snapshot

生命週期優於快照

Trust is not a point-in-time assertion. It is a lifecycle: require, generate, authorize, execute, evidence, coordinate, govern, audit. A snapshot proves a moment. A lifecycle proves a process. Systems that capture only the output miss the journey that created it.

信任不是時間點的斷言。它是一個生命週期：需求、生成、授權、執行、證據、協調、治理、稽核。快照證明一個時刻。生命週期證明一個過程。只捕獲輸出的系統錯過了創造它的旅程。

10. Correction over concealment

更正優於掩蓋

Every issuer of trust evidence will eventually be wrong — about a label, a claim, an implementation. What separates trust infrastructure from trust theater is what happens next: errors must be publicly disclosed, precisely scoped, and cryptographically linked to the artifacts they affect. An erratum is itself a trust object. A provider that has never published a correction has not yet been tested.

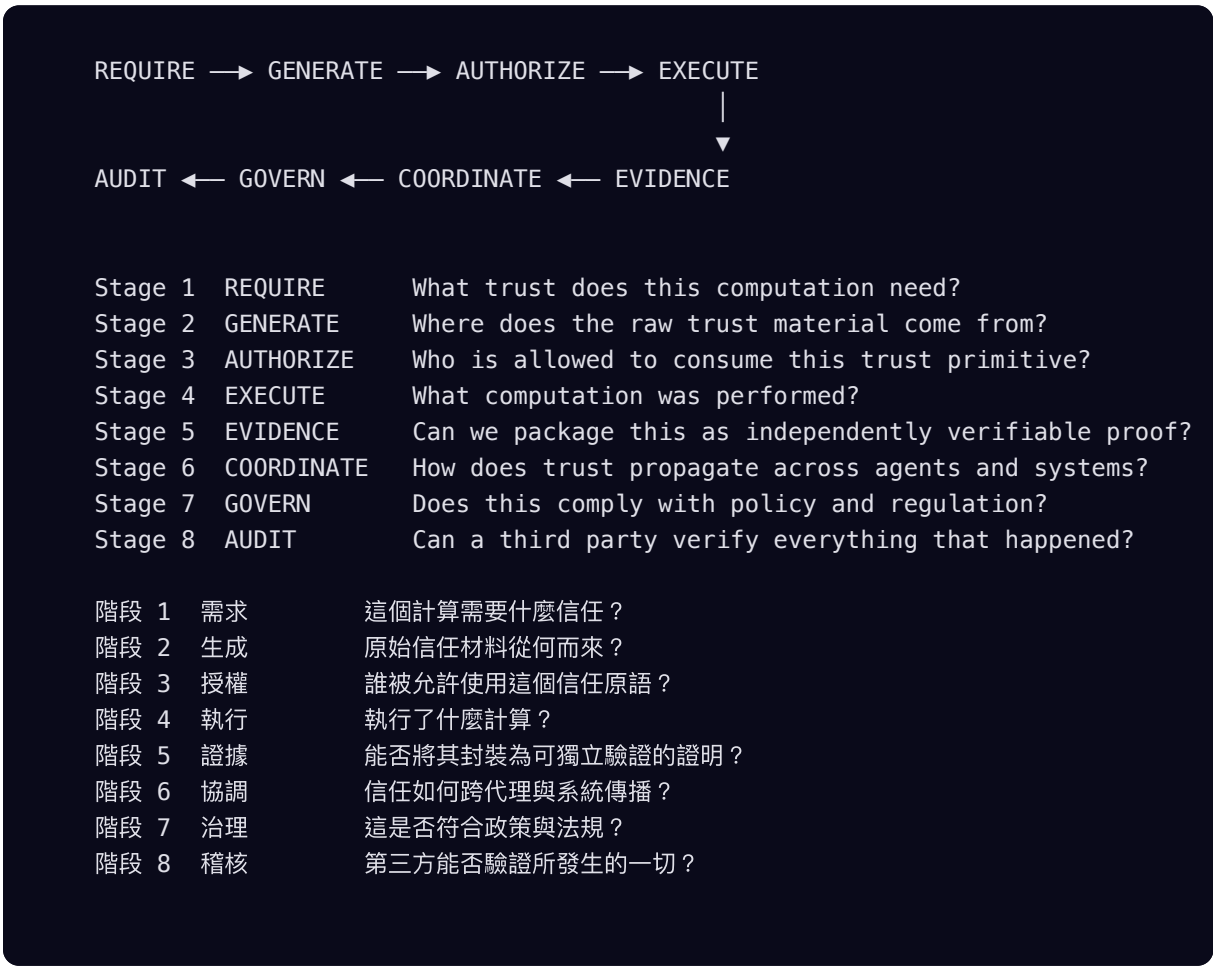
每個信任證據的發行者終將出錯——關於一個標籤、一個聲明、一個實作。區分信任基礎設施與信任表演的，是出錯之後的作為：錯誤必須公開揭露、精確界定範圍、並以密碼學方式連結到受影響的成品。勘誤本身就是一個信任物件。從未發布過更正的供應商，只是尚未被考驗。

The Trust Lifecycle

信任生命週期

Every infrastructure category is defined by its lifecycle. TLS has the handshake. OAuth has the authorization flow. DNS has the resolution chain. Computational Trust has the Trust Lifecycle — eight stages that transform a trust requirement into independently verifiable evidence.

> 每個基礎設施類別都由其生命週期定義。TLS 有握手。OAuth 有授權流程。DNS 有解析鏈。計算信任有信任生命週期——將信任需求轉化為可獨立驗證證據的八個階段。



Not every use case requires all eight stages. A development environment may use three. A production AI agent may use five. A regulated casino may use six. A multi-agent autonomous system may use all eight. The lifecycle is truncatable — valid at every stopping point at or beyond the minimal three — and additive — consumers adopt more stages as their trust requirements grow.

> **並非每個使用案例都需要全部八個階段。**開發環境可能使用三個。生產 AI 代理可能使用五個。受監管的賭場可能使用六個。多代理自主系統可能使用全部八個。生命週期是可截斷的——在達到最小三階段後的每個停止點都有效——且是遞增式的——消費者隨著信任需求增長採用更多階段。

Seven Invariants · 七大不變量

Regardless of how many stages are used, these invariants always hold:

Every stage produces a verifiable artifact. Where a stage rests on an external claim — an entropy source, a policy decision — the claim itself is signed, timestamped, and attributable. There are no *anonymous* trust-me steps: every residual assumption has a name, a key, and a record.

No stage requires trusting the previous stage's operator *to relay evidence honestly* — each artifact carries its own proof of what was committed. What signatures cannot prove — the honesty of the original claim — is exactly what the trust score and independent anchoring exist to constrain.

The lifecycle is re-entrant. A trust object from one lifecycle can be input to another.

Partial lifecycles are valid. Three stages produce a valid, if minimal, trust assertion.

Trust does not amplify. Child trust objects cannot claim higher trust than their parents.

Metadata is append-only. Once produced, artifacts are immutable.

Time moves forward. Temporal ordering is verifiable relative to independent anchors — an artifact can prove it existed no later than its anchor, and sequence violations are detectable.

> 無論使用多少階段，這些不變量始終成立：
>
> 1. **每個階段都產出可驗證的成品。**當某個階段依賴外部聲明——熵源、政策決定——該聲明本身必須被簽名、附上時間戳且可歸責。沒有*匿名的*「請相信我」步驟：每個殘餘假設都有名字、有金鑰、有記錄。
> 2. **沒有階段需要信任前一階段的操作者**——*會誠實轉交證據*——每個成品攜帶其所承諾內容的證明。簽章所無法證明的——原始聲明本身的誠實性——正是信任分數與獨立錨定所要約束的。
> 3. **生命週期是可重入的。**一個生命週期的信任物件可以成為另一個的輸入。
> 4. **部分生命週期是有效的。**三個階段產生有效的（儘管最小的）信任斷言。
> 5. **信任不會放大。**子信任物件不能聲稱比其父代更高的信任。
> 6. **元數據是僅可追加的。**一旦產出，成品就是不可變的。
> 7. **時間向前移動。**時間順序相對於獨立錨點可驗證——成品能證明其存在不晚於其錨點，且順序違規可被偵測。

The Gap

缺口

Computational Trust is not a replacement for existing infrastructure. It is the missing layer that connects existing infrastructure into a verifiable whole.

TLS secures the channel. OAuth controls who enters. TEE protects the execution environment. Blockchain provides immutable anchoring. Audit logs record what happened. Each does its job well. None of them proves *what actually happened* in the computational process — which inputs entered, which non-deterministic steps ran, in what order, under whose keys.

That is the gap. It is not a gap in any individual technology. It is a gap in the stack — a missing layer between "the system is secure" and "the process is trustworthy."

Computational Trust fills that gap: it proves **provenance and process integrity** — that the process ran as declared, tamper-evident and independently checkable. It does not certify that the outcome was wise or the model was right. Provenance, not correctness. That boundary is the discipline.

> 計算信任不是現有基礎設施的替代品。它是將現有基礎設施連接成可驗證整體的缺失層。
>
> TLS 保護通道。OAuth 控制誰進入。TEE 保護執行環境。區塊鏈提供不可變的錨定。稽核日誌記錄發生了什麼。每個都做好了自己的工作。它們都沒有證明計算過程中*實際發生了什麼*——哪些輸入進入、哪些非確定性步驟執行、以什麼順序、在誰的金鑰之下。
>
> 那就是缺口。它不是任何個別技術的缺口。它是堆疊中的缺口——「系統是安全的」和「過程是可信賴的」之間缺失的一層。
>
> 計算信任填補了那個缺口：它證明**出處與過程完整性**——過程確如宣告般執行，防竄改且可獨立查驗。它不保證結果是明智的，也不保證模型是對的。出處，而非正確性。這條界線正是這門紀律之所在。

The Pattern

歷史模式

Infrastructure categories are not invented. They are named.

They emerge when a problem becomes so pervasive that ad-hoc solutions can no longer contain it. Someone names the problem. Someone defines a standard. The standard becomes invisible infrastructure.

Problem	Name	Standard	Invisible Infra
Secure communication	TLS	RFC 5246	HTTPS everywhere
Delegated authorization	OAuth	RFC 6749	"Sign in with G
Portable identity	JWT	RFC 7519	Every API speaks
Container orchestration	Kubernetes	CNCF spec	Default deployme
Observability	OpenTelemetry	OTLP	Default telemetr
Verifiable computational provenance	Computational Trust	?	?

Distinct from zero-knowledge proof systems, which prove that a deterministic circuit executed correctly. Computational Trust proves what fed a process and what came out — across non-deterministic, multi-party lifecycles. The two are complementary.

> *有別於零知識證明系統——後者證明一個確定性電路被正確執行。計算信任證明的是過程的輸入與產出——跨越非確定性、多方參與的生命週期。兩者互補。*

The problem exists. Autonomous systems are making unverifiable decisions at global scale. The ad-hoc solutions are fragmenting — every enterprise is building custom audit trails that do not interoperate. The regulatory demand is converging — every major jurisdiction is independently requiring AI auditability without specifying how.

The name is here. **Computational Trust.**

The standard is what we are building.

> 基礎設施類別不是被發明的。它們是被命名的。
>
> 它們在問題變得如此普遍以至於臨時解決方案無法再應對時出現。某人命名問題。某人定義標準。標準成為隱形基礎設施。
>
> 問題已經存在。自主系統正在全球規模上做出不可驗證的決策。臨時解決方案正在碎片化——每個企業都在建構不能互操作的自定稽核軌跡。監管需求正在趨同——每個主要司法管轄區都在獨立要求 AI 可稽核性，卻沒有指定如何做到。
>
> 名稱已經在這裡。**計算信任。**
>
> 標準就是我們正在建構的。

The Invitation

邀請

This manifesto is not a product announcement. It does not belong to any single company, project, or implementation. It belongs to a belief: that autonomous computation requires a new trust layer, and that this layer must be open, standard, and universal. This manifesto is published by FairSeal but does not belong to it — the category outlives any company.

If you are building AI agents that make consequential decisions — you need Computational Trust.

If you are a regulator writing AI governance frameworks — you are describing Computational Trust, even if you do not yet have the name.

If you are an enterprise deploying autonomous systems — you will be asked to prove that those systems are trustworthy. Custom audit logs will not be enough.

If you are a developer building the next generation of autonomous software — your users will expect verifiability. It should be as invisible as HTTPS.

If you are building for the machine-to-machine economy — where agents transact with agents, pay for services, and consume each other's outputs with no human in the loop — you need trust objects that neither counterparty's operator controls. Payment rails for machines already exist. The receipt layer is what's missing.

We invite you to build with us. Not to adopt a product, but to help define a category. The standard does not yet exist. The specification is being written. The reference implementation is being built. Patents are being filed — defensively, and with the intent to license them royalty-free to any conforming implementation of the open standard — so this layer cannot be enclosed, by us or by anyone else. The open standard is the destination.

Computational Trust is not a feature. It is not a product. It is not a company.

It is the missing layer.

And it is time to build it.

> 這份宣言不是產品發布。它不屬於任何單一公司、專案或實作。它屬於一個信念：自主計算需要一個新的信任層，而這個層必須是開放的、標準的和普遍的。這份宣言由 FairSeal 發布，但不屬於它——類別的生命長於任何一家公司。
>
> 如果你正在建構做出重大決策的 AI 代理——你需要計算信任。
>
> 如果你是正在撰寫 AI 治理框架的監管者——你正在描述計算信任，即使你還沒有這個名稱。
>
> 如果你是正在部署自主系統的企業——你將被要求證明這些系統是可信賴的。自定稽核日誌將不夠。
>
> 如果你是正在建構下一代自主軟體的開發者——你的用戶將期望可驗證性。它應該像 HTTPS 一樣隱形。
>
> 如果你正在為機器對機器經濟建構——代理與代理交易、為服務付費、在無人參與的情況下使用彼此的產出——你需要任何一方操作者都無法控制的信任物件。機器的支付軌道已經存在。缺的是收據層。
>
> 我們邀請你與我們一起建構。不是採用一個產品，而是幫助定義一個類別。標準尚不存在。規範正在撰寫。參考實作正在建構。專利正在申請——以防禦為目的，並有意將其免權利金授權給開放標準的任何符合性實作——使這一層無法被圈占，無論是我們，或任何其他人。開放標準是目的地。
>
> 計算信任不是功能。不是產品。不是公司。
>
> 它是缺失的那一層。
>
> 是時候建構它了。

The Computational Trust Manifesto · 計算信任宣言

v2 — September 2026 · Published by FairSeal · fairseal.io/manifesto

v2, September 2026 — revised after our own corrections taught us the difference between provenance and correctness.

第二版，2026 年 9 月——在我們自身的更正教會我們「出處」與「正確性」之別後修訂。