

Trust State Protocol

A Context-Bound Model for Trust in Digital Systems

Karim Lahnin

www.karimlahnin.com

Expanded technical draft | June 2022

Abstract. Digital systems often express trust through ratings, static labels, or private scoring rules whose meaning is difficult to reproduce. We propose Trust State Protocol (TSP), a model that represents trust as a bounded state derived from confirmed outcomes within a declared context. Each accepted event changes the state through a weighted update; between events, the state moves toward a baseline as evidence ages. A context binds the outcome rules, verification requirements, event ordering, and numerical parameters needed to reproduce the result. The model separates evidence acceptance from score calculation and separates both from application decisions. This paper develops the original framework into a proposed processing profile with duplicate prevention, authenticated events, reproducible state transitions, and explicit limits. TSP does not establish the truth of an outcome, provide distributed consensus, or make trust a transferable asset. Its contribution is a common way to compute and interpret a time-sensitive trust signal under stated assumptions.

Introduction

A digital interaction can establish that a payment settled, a service was delivered, or a software agent completed a task. Yet the meaning of that outcome is often lost inside a platform-specific rating or a permanent account label. A reader cannot easily tell which events contributed, whether old evidence still matters, or why two systems produce different results from similar histories.

We need a model in which the meaning of a trust state follows from published rules. The state should concern a specific kind of interaction, change when an outcome is resolved, and become less informative when no new evidence arrives. It should also be possible to reproduce its calculation without treating the score as a statement about a person's overall character.

TSP addresses the calculation layer. Verification determines which outcomes may enter; TSP determines how accepted outcomes affect a bounded value; an application decides how to use that value. A high score does not authorize a transaction by itself. A low score does not constitute a finding of misconduct. Neither score is a calibrated probability of future success.

The original framework supplies the context-bound state, weighted transition, and baseline decay. The event envelope, processing profile, evidence-support measure, and correction procedure developed here are proposed extensions. They require implementation and evaluation before a production specification can claim interoperability or operational security.

Contexts and Trust States

An entity is any participant to which a state can be attached: a person, an organization, a device, or an automated agent. Its identifier need not reveal a legal identity. A context defines the particular outcome being measured. Payment settlement, delivery completion, and task execution are different contexts even when the same entity participates in all three.

A context manifest declares the protocol revision, context identifier, allowed score range, initial baseline, decay rule, event types, impact mapping, weight calculation, verifier authorization, and ordering rules. A conforming profile must also declare time units, rounding, and the treatment of disputes and corrections. Publishing a formula without these choices does not define a reproducible system.

$$T_{min} \leq T_{c,u}(t) \leq T_{max}$$

Here c identifies the context, u identifies the entity within that context, and t is the evaluation time. The baseline T_0 must lie inside the same range. A newly created state starts at T_0 ; this expresses the selected starting uncertainty, rather than a presumption of reliability or unreliability.

State component	Meaning
Context and manifest digest	The exact rules under which the state was computed.
Entity identifier	A context-specific reference to the participant.
Trust value and state time	The stored score and the time through which it was updated.
Last accepted sequence	The position of the latest event in the ordered stream.
History commitment	An optional digest linking the accepted event history.
Evidence support	Optional decayed support, distinct from the score.

Contexts are separate namespaces. A trust value of 80 in one context cannot be interpreted as 80 in another without examining both manifests. A delivery record does not establish payment reliability, and a payment record does not establish competence to perform a technical task. The core model has no operation for transferring or averaging scores across contexts.

Separation is a rule for interpretation, not a guarantee against external data linkage. An operator can still correlate identifiers or create an additional model using several contexts. Such a model falls outside the context-bound calculation described here and must state its own assumptions.

A context may be defined by one operator or agreed upon by several participants. The mathematical model does not require a public blockchain, a token, or a global registry. A shared deployment nevertheless requires agreement on who may confirm events and which history is authoritative. Those choices belong to the deployment profile.

Confirmed Events

An event represents a resolved outcome, rather than an intention or an unverified claim. Starting a payment is not settlement. Submitting a task is not acceptance of its result. The manifest must define the point at which an outcome becomes eligible, including any settlement delay, dispute window, or required independent evidence.

The proposed event envelope contains a context identifier, manifest digest, entity identifier, unique event identifier, interaction identifier, outcome code, occurrence time, confirmation time, ordering sequence, and verifier reference. It may also contain an evidence commitment and an authentication proof. The outcome and any authenticated factual attributes are mapped to an impact and weight by the manifest; an untrusted sender cannot choose its own score increase.

An interaction identifier prevents one underlying result from being represented as several independent successes. An event identifier detects exact duplicates. These controls answer different questions: two differently encoded messages can describe the same interaction, while one valid message may be delivered repeatedly. Both checks are necessary.

Before an event affects a state, the processor checks that its context and manifest match, its verifier is authorized, its authentication proof is valid, its outcome meets the confirmation criteria, and its identifiers have not already been consumed. It then checks the allowed age, timestamps, sequence, and parameter bounds. Pending, rejected, and disputed outcomes do not enter the score until the stated resolution rule permits them.

Verification and Its Limits

Confirmation means that the declared acceptance process has been satisfied. It does not mean that the underlying account of reality is infallible. A signature identifies the key that authenticated a message and protects its integrity; it does not prove that a service occurred or that a verifier was honest. Mutual agreement can also be fabricated by cooperating participants.

A payment context might accept a signed settlement receipt from an authorized service. A task context might require an independently checked output. A delivery context might require a recipient confirmation combined with other evidence. These are examples of context design, not interchangeable guarantees.

The manifest should state what happens when a verifier key is revoked, when an outcome is challenged, and when a required verifier is unavailable. Revocation of a key does not automatically determine whether all earlier events were false. Historical corrections require a declared rule and enough retained information to apply it.

Verification strength controls admission or feeds a published weight rule; it is not an undocumented adjustment made after calculation. Replicas need the same accepted outcomes and authenticated inputs to reproduce the same state. A deployment that cannot resolve contradictory confirmations must expose that uncertainty instead of publishing a falsely definitive score.

Time and State Transitions

Let T_{i-1} be the stored value immediately after the preceding accepted event at time t_{i-1} . Before accepting the next event at confirmation time t_i , the processor evaluates the old state at that time. With a strictly positive decay rate λ , the decayed value B_i is:

$$B_i = T_0 + (T_{i-1} - T_0) \exp[-\lambda (t_i - t_{i-1})]$$

The accepted event then applies the original weighted transition:

$$T_i = (1 - a_i) B_i + a_i I_i, \quad 0 \leq a_i \leq 1$$

I_i is the event target, not an amount added to the score. The weight a_i determines how far the state moves toward that target. A small weight preserves more of the prior state; a weight of one replaces the decayed value with the target. The manifest must constrain I_i to the allowed score range.

For an illustrative directional profile, confirmed success targets $T_{\max}$ and confirmed failure targets $T_{\min}$. A neutral event has weight zero. This definition avoids a contradiction in which an event described as positive lowers a score already above its target. Other profiles may use intermediate targets, but must describe the resulting movement accurately. An event always moves toward its target, not necessarily upward.

Neutrality means no event-driven score change. The time passage preceding a neutral event can still change the evaluated value through decay. A neutral event may advance the stream position and state time, but must not be counted as new score-bearing evidence unless a separate evidence rule expressly defines that treatment.

Baseline Decay

Decay reduces the distance from the baseline. A value above T_0 falls; a value below T_0 rises. Inactivity therefore does not always reduce the numerical score. It reduces the persistence of both favorable and unfavorable history. This is a substantive choice: contexts needing durable restrictions should retain those restrictions separately from a decaying trust signal.

$$\lambda = \ln(2) / h$$

The half-life h is the time required to halve the distance from the baseline. It must use the same unit as the timestamps. At an arbitrary read time, the processor applies decay from the stored state time without inserting an event. Reads should not repeatedly round and rewrite the stored value; otherwise the answer may depend on how often the state was queried.

Confirmation time is authoritative in this proposed profile. Occurrence time supports age checks and audit. Decay does not prevent delayed-report manipulation by itself: the manifest also needs admissible reporting delays, trusted time rules, and a policy for events that arrive after the accepted sequence has advanced.

Ordering, Replay, and Corrections

The update is order-sensitive. Starting at 50 with weight 0.2, a success toward 100 followed immediately by a failure toward 0 produces 48. Reversing those events produces 52. The same set of outcomes is therefore insufficient: replicas must agree on an ordered event stream and the associated confirmation times.

A centralized profile can assign a monotonically increasing sequence per entity and context when admitting events. Confirmation times must be nondecreasing along that sequence. Equal timestamps are processed in sequence order. Independent arrival order or a sender-selected timestamp is not a safe substitute for this agreement.

The score update, stream advancement, history commitment, and consumed-identifier records must be committed atomically. If the process crashes, a retry must either apply the whole event once or recognize that it has already been applied. Concurrent processors need serialization or an equivalent compare-and-swap mechanism to avoid overwriting each other.

A replayed event is rejected through the consumed-event and interaction records, not through decay. These records must survive ordinary restarts and history compaction for the relevant replay horizon. A deployment may replace a full history with an authenticated checkpoint, but it still needs a duplicate-prevention mechanism covering events that remain admissible.

Proposed Processing Procedure

```
load manifest and stored state
validate envelope, verifier, outcome, age, and bounds
reject consumed event or conflicting interaction result
require the next authoritative sequence and valid time
compute decay from stored time to confirmation time
derive target and weight from the manifest
compute and quantize the new state
atomically commit state, sequence, and consumed records
```

This procedure specifies the logical order of work. Authentication and storage are supplied by the deployment. A networked system must additionally define how admission decisions and sequence assignments become final across replicas. TSP does not supply a consensus algorithm or a fork-choice rule.

Correcting an Accepted Outcome

An accepted event is not silently overwritten. A correction record identifies the affected interaction, provides authorized replacement evidence, and states its effective treatment. One possible profile rebuilds the state from a checkpoint before the affected event, substitutes the corrected outcome at that original stream position, and replays all later events under the same historical rules. The rebuilt result receives a new revision and commitment.

Simply adding an opposite event is generally not an inverse: the update depends on order, weight, and intervening decay. If historical replay is unavailable, a prospective adjustment can be defined, but must be labeled as that policy rather than exact restoration. A system that deletes its replay material must disclose the resulting limit on correction and audit.

Calculations and Evidence Support

Consider a context with range 0 to 100, baseline 50, half-life 30 days, success target 100, and failure target 0. Success weight is 0.2 and failure weight is 0.4. These parameters illustrate the mechanics; they have not been calibrated against any deployment or loss data.

Time	Accepted outcome or read	Calculation	Trust
Day 0	New state	Baseline	50.000
Day 0	Confirmed success	$0.8 \times 50 + 0.2 \times 100$	60.000
Day 30	Before next event	$50 + (60 - 50) \times 0.5$	55.000
Day 30	Confirmed failure	$0.6 \times 55 + 0.4 \times 0$	33.000
Day 60	Read; no new outcome	$50 + (33 - 50) \times 0.5$	41.500

The failure has more immediate influence because its declared weight is larger. The later increase from 33 to 41.5 is not evidence of improved behavior. It represents movement toward the chosen uncertainty baseline as the unfavorable observation becomes older.

A score alone also hides how much evidence supports it. A new entity at 50 and an entity returning toward 50 after many events have different histories. The proposed optional evidence-support measure N records decayed support separately:

$$N_i = \exp[-\mu (t_i - t_{i-1})] N_{i-1} + v_i$$

N starts at zero. The positive rate μ governs support decay, while v_i is a nonnegative contribution assigned by the evidence rule. A simple profile uses one for each eligible, distinct, score-bearing interaction and zero for neutral events. Correlated interactions may need reduced or capped contributions. N is a support indicator, not a probability or statistical confidence interval.

Repeated Outcomes

For consecutive identical targets I, constant weight a, and no elapsed time, the recurrence has a closed form:

$$T_n = I + (T_{start} - I)(1 - a)^n$$

With the example success target and starting value, ten successes yield approximately 94.631. Even a small positive weight can approach the target after enough repetitions. Boundedness prevents values outside the permitted range; it does not prevent artificial success farming or establish that each event supplies independent evidence.

A context can limit contributions from repeated counterparties, require independently confirmed outcomes, or cap eligible activity within a declared period. Those rules must be applied before or during the deterministic update, and must define how legitimate repeated activity is treated. Their effectiveness is an empirical question. The recurrence itself cannot distinguish honest volume from collusive volume.

Properties and Security Assumptions

The model has useful mathematical properties under explicit conditions. Since $\exp(-\lambda dt)$ lies between zero and one for nonnegative elapsed time, decay is a convex combination of the old value and the baseline. The event update is also a convex combination when its weight lies between zero and one. If all inputs remain within the permitted range, both operations preserve that range.

For two starting scores processed with the same event inputs, the difference after one transition is multiplied by $(1 - a_i) \exp(-\lambda dt)$. The multiplier lies between zero and one. Earlier score differences therefore cannot expand under identical accepted inputs. This contraction is a property of the recurrence, not proof that an attacker cannot manipulate the accepted inputs.

$$|T_i - T'_i| = (1 - a_i) \exp(-\lambda dt) |T_{i-1} - T'_{i-1}|$$

Decay can be evaluated lazily because two exact decay intervals combine into one interval of their total duration. Rounding can disturb that equality, which is another reason not to persist a new rounded state after every read. Identical event histories yield identical logical results only when the manifest, ordering, timestamps, and arithmetic rules also agree.

Adversarial Behavior

Collusion and Sybil identities. Participants may create multiple accounts or cooperate to fabricate qualifying outcomes. TSP does not provide identity uniqueness or Sybil resistance. Identity binding, costly eligibility, verifier independence, or limits on correlated activity can be added by a context, but each introduces assumptions, expenses, and potential barriers to legitimate users.

Account resets and transfers. Returning to the baseline can be attractive to an entity with an unfavorable history. A newly registered identifier is not proof of a new participant. Account continuity and credential-transfer rules must be supplied separately. A stolen or sold account also invalidates the assumption that past outcomes concern its current controller.

Verifier compromise and selective reporting. An authorized verifier can sign false events, withhold failures, or delay publication. Signatures cannot cure this. The deployment needs declared authorization, review, key-revocation, and reporting rules. It should assess whether evidence producers have incentives to submit both favorable and unfavorable results.

History substitution and timing attacks. An operator may omit events, backdate confirmations, or show different histories to different readers. Hash commitments can make alterations detectable relative to a trusted checkpoint; they cannot by themselves prove completeness or prevent competing histories. Trusted time, authoritative ordering, and replication agreement remain external requirements.

No numerical score guarantees honest future conduct. An attacker may build a valid history and later exploit it in a larger transaction. Applications must control exposure using additional evidence and transaction-specific limits. The costs and benefits of manufacturing qualifying events must be evaluated in each deployment.

Privacy and Auditability

A score processor can operate on normalized outcome records rather than raw messages, identity documents, or continuous behavioral monitoring. The verifier may still need sensitive source material. Separating those roles can reduce what the scoring service stores, but it does not eliminate the verifier's data obligations or the sensitivity of the resulting state.

Context-specific pseudonyms reduce routine linkage when the mapping is protected and identifiers are not reused. They do not provide anonymity by themselves. Event timing, interaction relationships, and distinctive score changes can reveal activity. Frequent public snapshots may expose more than a deployment intends, even when names are absent.

A history commitment can bind a sequence of normalized records. For example, a profile may compute H_i from a domain label, context, manifest digest, prior commitment, sequence, and digest of the accepted envelope. Canonical encoding is required so that equivalent records have the same bytes. RFC 8785 is one available JSON canonicalization scheme; its adoption would be a profile choice (Rundgren et al., 2020).

A digest protects a commitment to bytes, not their truth. Hashing a predictable private fact may permit guessing attacks. Private evidence commitments need appropriate randomization and access controls. Public disclosure of event hashes or Merkle proofs also requires analysis of what those objects reveal.

An authorized auditor with the ordered accepted records and manifest can recompute the state and compare it with a snapshot. A reader holding only the snapshot must trust its issuer or use an additional proof system. Selective disclosure, zero-knowledge proofs, and trusted execution are possible extensions; this paper does not define or claim them.

Storage and Interoperability

A current-state read and a single update require constant arithmetic work. Duplicate records, accepted history, verifier registries, and correction material require additional storage. Full replay is linear in the number of accepted events being replayed. Per-entity ordering also constrains parallel processing within one stream, although independent entity-context streams can be processed separately.

An export should bind the entity, context, manifest digest, protocol revision, trust value, evidence support if used, evaluation time, sequence, and history commitment. A digital signature may authenticate that export. Ed25519, specified in RFC 8032, is one possible profile choice; it authenticates the signed statement without validating the underlying outcome (Josefsson and Liusvaara, 2017).

Numerical interoperability needs more than an agreement to use exponential decay. A profile must define precision, overflow behavior, parameter encoding, rounding points, and either a deterministic exponential routine or a published fixed-point approximation. Test vectors must cover boundaries, equal timestamps, neutral events, retries, and corrections. Floating-point libraries with unspecified rounding are insufficient for a bit-identical guarantee.

Deleting old evidence does not erase its influence from a checkpointed score. Retention and deletion rules therefore need to address records, derived states, commitments, backups, and the ability to revise them. TSP specifies no legal compliance guarantee and should be assessed against the obligations of its actual deployment.

Deployment, Versioning, and Evaluation

TSP can support recurring interactions in marketplaces, settlement services, delivery networks, and automated task systems when outcomes can be resolved under declared rules. It can also be implemented inside regulated environments as a limited analytical component; it does not replace their applicable verification, review, or compliance requirements. A context lacking reliable outcome evidence gains little from a precise scoring formula.

The least complex deployment is a single service with an authenticated event source, an ordered append log, and transactional state storage. A federation additionally needs agreement about verifier membership, admission, sequencing, and conflict resolution. A public permissionless deployment would require its own evidence, consensus, and adversarial participation model. TSP does not resolve those requirements by being described as a protocol.

Every exported state should identify the exact manifest and revision. A change to weights, targets, baseline, decay, outcome rules, or arithmetic can change meaning even if the score range stays the same. Historical replay uses the historical rules. A migration must declare whether it starts a new context, rebuilds eligible history under new rules, or establishes an explicitly mapped starting state.

Before production use, a reference implementation should publish reproducible test vectors and evaluate parameter sensitivity. Simulation should examine sparse histories, bursts, clustered counterparties, withheld failures, delayed events, key compromise, account resets, and correction replay. Outcome data should be used to measure usefulness and error patterns. No implementation benchmarks, empirical calibration, or independent security validation are reported in this draft.

Conclusion

We have proposed a model for expressing trust as a bounded, time-sensitive state within a declared context. Confirmed outcomes supply event targets and weights; baseline decay reduces the persistence of older evidence; an ordered history makes the calculation reproducible. The expanded profile identifies the event, storage, and verification rules that a working deployment must add to the original recurrence.

The result is a common calculation discipline with explicit boundaries. Its credibility depends on the evidence entering it, the integrity of the accepted history, and the care with which applications interpret the signal. TSP supplies neither universal reputation nor a replacement for identity, consensus, or judgment. Further work should establish a reference profile, publish test vectors, and test the model against real outcomes.

References

Lahnin, Karim. *Trust State Protocol (TSP): A Framework for Trust Mechanics in Digital Systems*. June 2026. Author-supplied original draft; basis for the state model and this expanded paper.

Nakamoto, Satoshi. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. Author-supplied copy. Reference for technical-paper presentation; TSP does not adopt Bitcoin's proof-of-work or consensus model.

Rundgren, Anders; Jordan, Bret; Erdtman, Samuel. *JSON Canonicalization Scheme (JCS)*. RFC 8785, June 2020.
<https://www.rfc-editor.org/rfc/rfc8785>

Josefsson, Simon; Liusvaara, Ilari. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032, January 2017.
<https://www.rfc-editor.org/rfc/rfc8032>

Authorship: Karim Lahnin. This expanded draft develops the author's original framework and identifies proposed additions. It is an independent technical proposal, not an established standard. Contact: www.karimlahnin.com.