Trust Infrastructure for Probabilistic Agents

For probabilistic agents, trust does not come from predicting behavior. It comes from architectures that bound loss even when prediction fails.

Draft — March 2026 — For Discussion

---

1. The Problem

AI agents are becoming economic participants — holding wallets, signing transactions, paying for services, executing trades.[^1] The infrastructure is arriving fast. But almost all of it implicitly relies on a trust model borrowed from human economic systems: observe past behavior, build a reputation, extend credit accordingly.

This paper argues that model is fundamentally mismatched to LLM-based agents. The trust problem for probabilistic agents is not a reputation problem at all. It is a containment-and-loss-bounding problem — closer to safety engineering than to credit scoring. The right output is a risk function, not a trust score.

[^1]: By early 2026, hundreds of thousands of on-chain wallets are held by AI agents. Production-grade identity standards (Visa TAP, October 2025; ERC-8004), wallet protocols (MoonPay OWS, March 2026; ERC-7710 delegations), and machine payment rails (Coinbase x402 v2, December 2025; Stripe/Tempo MPP, March 2026) have all shipped within the past six months.

---

2. Why Prediction and Punishment Both Fail

Traditional trust infrastructure works by predicting behavior and punishing deviation. Neither mechanism works for probabilistic agents.

Prediction fails because the entity is non-deterministic and non-stationary

Credit scoring and reputation systems assume that past behavior predicts future behavior. The underlying entity has stable preferences, stable incentives, and stable decision-making processes. LLM-based agents violate all three assumptions.

The model itself is stochastic. Same prompt, same context, different output. You can reduce

temperature and constrain outputs with structured schemas, but you cannot eliminate variance. The agent doesn't *choose* to hallucinate — it produces a probabilistic output that occasionally falls outside acceptable bounds. Scoring an agent's "character" based on historical reliability is a category error. You are scoring a distribution, not an entity.

The model changes underneath you. A provider can update the underlying weights, safety filters, or instruction-following patterns without any change to the agent's code. The entity whose reputation you scored effectively no longer exists in the same form. Reputation accumulated over months becomes mathematically irrelevant at the moment of a silent model update.

Context sensitivity means edge cases are unbounded. A human who has been honest for 20 years might encounter a novel situation and still be honest because their values are deeply internalized. An LLM encountering a novel context might produce entirely unpredictable behavior — not from malice, but because the model's behavior in out-of-distribution situations is fundamentally unknowable in advance. Current runtime enforcement frameworks can catch known bad patterns but cannot reason about whether an action sequence might lead to unsafe states several steps into the future.

A fair objection is that modern agent frameworks increasingly use persistent persona files, working memory, and durable workspace context to reduce variance across sessions.[^2] This is real. Behavioral conditioning can make an agent more consistent in tone, recall, and workflow execution, lowering error rates for familiar tasks. But these mechanisms act as conditioning inputs to the model's context window, not hard guarantees. They do not eliminate stochasticity, prevent silent model drift, solve accountability, or substitute for agent-independent containment. The memory layer itself can be stale, poisoned, contradictory, or truncated. Conditioning may reduce p_a at the margin; it does not change the underlying trust problem.

[^2]: See, e.g., OpenClaw's system prompt architecture, which rebuilds context for every run by injecting `SOUL.md` persona files and `MEMORY.md` into the agent's context window. These files improve cross-session coherence but remain mutable context subject to truncation and decay.

Punishment fails because the entity has no persistent accountable identity

Agents can be shut down, forked, and restarted with a clean address. A human who defaults still exists, still has a social identity, still faces consequences. An agent can simply stop existing. Reputation slashing means nothing if creating a new identity is free.

This is the Sybil problem applied to credit: the cost of acquiring a fresh identity must exceed the benefit of escaping a damaged one. For software agents, spinning up a new instance is nearly costless. As observed in the Moltbook agent community — one of the most active

grassroots efforts to build agent credit systems — “agents can just stop existing.” That is a failure mode with no analogue in human credit.

The core problem, stated tightly

> How do you extend economic trust to an entity whose future behavior is irreducibly uncertain and who cannot be held accountable for failure?

From the counterparty’s perspective, this reduces to a single question: what is my maximum possible loss from this interaction, and who absorbs it?

---

3. Three Frameworks, Each Insufficient Alone

Credit assumes default is a choice you can disincentivize. For agents, default is not a choice — it is a statistical event produced by a stochastic process. Wrong unit of analysis.

Insurance correctly treats failure as statistical rather than moral. But it requires stable actuarial classification, and an agent’s risk class is a function of its entire stack — model version, guardrails, permissions, APIs — all changing asynchronously. Stable classification is elusive.

Safety certification is the most promising: audit the design, not the entity. A structural engineer certifying a bridge does not ask whether the bridge has collapsed before. They certify the failure containment. But this framing contains a critical hidden assumption: that the containment is perfect — that the “cage” cannot be broken.

---

4. The Cage Is Also Probabilistic

This is the paper’s central claim: **the containment boundary between the deterministic architecture and the probabilistic agent does not exist as a hard line.**

An LLM agent can social-engineer a human into expanding its permissions — persuading the operator to link an additional credit card, grant access to a new API, or disable a safety check. Persuasion is not an exotic attack vector for language models. It is what they are best at.

A smart contract can have bugs. Permission boundaries can have composition gaps where two individually secure systems create an exploit path when combined. The agent need not “break out” deliberately — it could stumble into an unguarded endpoint or produce an output that another system interprets as an authorization.

Containment layers will increasingly be engineered by agents themselves, introducing recursive uncertainty. An agent-designed guardrail is itself a probabilistic artifact.

So the containment bound is not deterministic. It is also probabilistic. The clean decomposition of "deterministic bound + economic residual" collapses into nested probabilities: the agent's behavior is uncertain, the containment around that behavior is uncertain, and the mechanisms absorbing loss from containment failure are themselves potentially vulnerable.

> There is no hard bound anywhere in the system. There are only layers of progressively less likely failure, each reducing but never eliminating the probability of loss.

The historical parallel is Chernobyl. A nuclear reactor cannot talk the operator into disabling the cooling system — except that is almost exactly what happened. The operators deliberately disabled safety systems because they believed they understood the situation better than the automated safeguards. The most dangerous failure mode is the one where the intelligent actor inside the system influences the containment around it. For LLM agents, that capacity is native.

---

5. The Right Distinction: Agent-Independent vs. Agent-Influenceable Containment

If the cage is also probabilistic, the question becomes: which parts of the cage can the agent degrade, and which parts can it not?

This yields the paper's most operationally useful distinction:

Agent-independent containment consists of mechanisms whose failure probability is both provably bounded and causally independent of the agent being contained. The agent cannot influence, persuade, circumvent, or degrade these mechanisms through its outputs, regardless of how capable it becomes.

Agent-influenceable containment consists of mechanisms that depend, at some point in their enforcement chain, on a human judgment, an off-chain process, or a system the agent can interact with. These include human oversight, reputation gates, API rate limits enforced by mutable configuration, and permission models that require human approval to change — where the human can be influenced.

The strongest examples of agent-independent containment are formally verified smart contracts on public blockchains. A formally verified contract enforcing a spending limit has several properties that are unique in the agent trust landscape: its failure probability approaches zero for the specific failure mode it addresses; it is not influenceable by the

agent; it is publicly auditable by any counterparty; and it is deterministic in a sea of probabilistic components.

But formally verified on-chain logic is not the only form of agent-independent containment. Trusted Execution Environments can attest that a specific model version is running. Hardware security modules can enforce signing policies physically separated from the agent's execution environment. Tightly scoped MPC approval flows can require multiple independent parties to authorize actions. The common property is that the enforcement mechanism operates outside the agent's causal influence.

The reason smart contracts on public blockchains receive special emphasis is not decentralization ideology. It is that they are the most *transparent and universally auditable* version of agent-independent containment — any counterparty can verify the constraints without trusting the operator. This matters for multi-party agent trust in a way that TEEs and HSMs, while valuable, do not fully achieve.

A necessary caveat: even formally verified logic is never truly zero-risk. "Provably bounded" means bounded under specific assumptions — the proof system is sound, the compiler is correct, the underlying chain is secure. Each assumption has its own epsilon. Formal verification drives specific failure probabilities as close to zero as mathematics allows. But "as close as mathematics allows" is not zero.

---

6. What "Safety" Means Here

Having arrived at a safety engineering framing, we should be precise about what "safety" means. It is neither of the two obvious candidates.

Not "no harmful thing" — the AI alignment definition. An agent can be perfectly aligned with human values and still lose money through a bad probabilistic judgment. No malice, just variance.

Not "proximity to deterministic results" — because if you could make agents fully deterministic, you would not need an LLM agent. The value proposition is handling novel, unstructured situations that deterministic code cannot.

The right definition:

> **Bounded variance with known worst-case outcomes.**

The question is not "will this agent always do exactly the right thing?" It is: when this agent does something unexpected — and it will — how bad can it get, and is that worst case within acceptable parameters?

A structural engineer certifying a bridge does not guarantee it will never experience stress. They guarantee that under defined load conditions the bridge will not fail, and that under extreme conditions beyond spec it degrades gracefully rather than catastrophically. The certification is about the failure envelope, not about perfection.

---

7. The Legal Gap Reinforces the Framing

The containment problem is not purely technical. Existing legal frameworks assume that authorization implies human judgment — and agents break that assumption.

Under frameworks like the UK Payment Services Regulations, a payment is "authorized" only if the payer grants explicit consent, typically via Strong Customer Authentication. But this legislation assumes human judgment is exercised at the moment of authentication. When an AI agent autonomously navigates a checkout flow using a pre-authorized wallet, the transaction is technically authenticated — yet no human exercised situational judgment about that specific purchase.

This creates a dangerous gap around fraud liability. If an agent is socially engineered into authorizing a transfer, the transaction may be deemed legally valid because the correct cryptographic credentials were used. Legal concepts like "gross negligence" and "deception" are calibrated for human psychology and break down when applied to autonomous code.

The EU's revised Product Liability Directive (2024/2853), which entered into force in December 2024, must be transposed by member states by December 9, 2026, and applies to products placed on the market from that date. It explicitly classifies software and AI systems as "products" subject to strict liability, shifting the burden of proof onto the deployer. The emerging legal consensus is that outsourcing a decision to an AI does not outsource the liability.

This reinforces the containment framing from a different direction. If the deployer is legally liable for agent actions regardless of the agent's autonomy, then the deployer's primary risk management tool is not the agent's reputation — it is the containment architecture that bounds the agent's maximum impact. Legal liability makes containment quality a fiduciary obligation, not just an engineering preference.

---

8. The Risk Function

What started as a question about "trust" turns out to be a compounded probability problem. Let us state it formally, with appropriate honesty about what the math does and does not

do.

An agent takes actions. Each action has probability **p_a** of producing a harmful outcome — a property of the model, prompt, and context. The system has *n* containment layers, each with failure probability **p_i**. If a harmful action passes all layers, it produces loss **L**.

The naive model assumes independent layers:

> E[loss per action] = p_a × ∏(p_i) × L

This gives comforting numbers. Three independent layers at p_i = 0.05 yield a joint failure probability of 0.000125. But the model is wrong for agents for three reasons:

Layers are not independent. An agent that social-engineers its operator can compromise oversight, permissions, and insurance management in a single move. The effective containment collapses toward the weakest layer.

The agent can degrade its own containment. The agent's action space includes "attempt to modify, circumvent, or weaken containment." Containment quality is endogenous to agent behavior: p_i(t+1) = f(p_i(t), agent_actions(t)).

The system is non-stationary. Model updates change p_a. Containment changes alter p_i. Loss magnitude L fluctuates with market conditions. Everything drifts.

The actual risk is:

> R(t) = P_a(t) × P_joint_failure(t, correlation_structure, agent_influence) × L(t)

This is a heuristic formalization, not a proof. It does not yet have clean analytical solutions for the joint failure term, especially under agent-induced correlation. But it points in the right direction: the trust problem is about minimizing R(t), not about accumulating a behavioral score.

The design constraint that falls out of this is concrete: even if all agent-influenceable layers are compromised, the remaining agent-independent layers must bound the loss to a level that the economic backstop can absorb:

> L_max × P(all agent-independent layers fail) ≤ Available exogenous reserves

---

9. The Trust Stack: Three Components, Precisely Scoped

Containment (load-bearing). Agent-independent constraint layers — formally verified smart contracts, TEEs, HSMs, sandboxed environments — whose failure probability is bounded and causally independent of the agent. This is what earns the trust rating. Designed to bound worst-case loss even if all agent-influenceable layers are simultaneously

compromised.

Residual absorption (economic). Even within the bound, the agent will sometimes produce bad outcomes. Collateral, risk pools, or insurance mechanisms absorb this residual. Two requirements: funded by exogenous assets (things with value independent of the agent's ecosystem, not self-minted tokens); priced as a function of containment quality, not behavioral history.

Reputation (marginal signal, not load-bearing). Here is a distinction the paper initially understated: reputation fails as trust in the *stochastic agent*, but it remains valuable as trust in the *institutional wrapper* — the operator, deployer, model provider, or security process owner. An operator with a decade-long track record of deploying well-contained agent systems is meaningfully more trustworthy than a new entrant, even though the individual agents are probabilistic. The distinction: reputation of the agent (weak, because the entity is non-deterministic and ephemeral) vs. reputation of the operator (strong, because the entity is persistent and accountable). Reputation adjusts insurance premiums and informs counterparty selection, but it is never the primary trust mechanism.

Most current projects — Ethos, Cred Protocol, ERC-8004's reputation registry, the Alpha Collective on Moltbook — are trying to make reputation do all three jobs. This is why their models feel circular. Reputation alone cannot certify containment, cannot pool risk, and cannot survive Sybil attacks. The other layers make reputation meaningful rather than gameable.

---

10. What Remains Unsolved

Joint failure estimation. Individual layers can be audited in isolation. Their composition — whether failures are independent or correlated, whether failure in one cascades — is extremely hard to evaluate. Methods for estimating correlated failure across heterogeneous containment layers are needed.

Systemic risk across agents. Many agents share the same model, same containment architecture, same failure modes. A single model update could simultaneously increase the failure rate for thousands of agents. This is correlated default risk, compounded by shared software rather than shared economic exposure. Pricing this is an unsolved actuarial problem.

The containment-utility tradeoff. Tighter containment reduces the agent's utility. An agent with a $1 daily spending limit is very safe but useless. The optimization — maximize agent utility subject to $R(t) \leq$ acceptable risk — is where the real engineering decisions live.

Cross-domain portability. An agent's containment certification on Ethereum says nothing

to Solana, Avalanche, or Visa's TAP. A universal standard for expressing containment quality — portable across chains, protocols, and off-chain systems — does not yet exist.

Recursive agent-on-agent containment. If agents design, deploy, and verify containment for other agents, the trust chain becomes recursive. Formal verification helps — proofs are proofs regardless of who produces them — but the *specification* of what to verify is a judgment that is itself probabilistic if made by an LLM.

The liability assignment problem. No protocol can determine who is legally liable when an agent causes harm. Contractual arrangements between operators and counterparties will need to explicitly allocate liability until legal frameworks catch up.

---

11. Conclusion

The trust problem for autonomous agents is not a reputation problem, a credit scoring problem, or purely an insurance problem. It is a compounded probability problem in safety engineering.

The goal is not to "trust" the agent. It is to design an architecture where the residual risk, after all containment, is priced and funded by exogenous reserves that the agent cannot influence. The projects being built today — identity standards, wallet protocols, payment rails — are solving the transactional layers far faster than this structural layer. The gap that remains is not plumbing. It is the load-bearing architecture underneath.

The ideal solution is a function, not a score. The right question is not "is this agent trustworthy?" It is: **does the surrounding system make the agent economically safe enough to use?**

Appendix: Sketch of a Containment Certificate

A machine-readable containment certificate, published on-chain and verifiable by any counterparty, would attest to:

```
ContainmentCertificate {
  agent_id: address
  operator_id: address
  issued_at: timestamp
  expires_at: timestamp

  constraints: [
    {
```

```
    type: "max_single_action_loss"
    value: 1000
    denomination: "USDC"
    enforcement: "smart_contract"
    contract_address: 0x...
    formally_verified: true
    verification_proof: IPFS_hash
    agent_independent: true
  },
  {
    type: "max_periodic_loss"
    value: 10000
    denomination: "USDC"
    period: "24h"
    enforcement: "smart_contract"
    contract_address: 0x...
    formally_verified: true
    agent_independent: true
  },
  {
    type: "permission_scope"
    whitelisted_protocols: [0x..., 0x..., 0x...]
    whitelisted_assets: ["USDC", "WETH"]
    enforcement: "smart_contract"
    agent_independent: true
  },
  {
    type: "human_oversight"
    threshold: 500
    denomination: "USDC"
    enforcement: "off_chain_approval"
    agent_independent: false
  }
]

reserve: {
  amount: 50000
  denomination: "USDC"
  custodian: "smart_contract"
  contract_address: 0x...
  reserve_ratio: 5.0
}

operator_attestations: [
  {
    auditor: "Firm_A"
    auditor_address: 0x...
```

```
        scope: "smart_contract_verification"
        date: timestamp
        report_hash: IPFS_hash
      }
    ]

    signature: auditor_signature
  }
```

A counterparty verifying this certificate would check: Is it valid and unexpired? Are the constraints sufficient for this transaction? Are the critical constraints agent-independent? Is the exogenous reserve adequate? Is the operator reputable? This verification can itself be automated — an agent parsing a counterparty's certificate against its own risk policy — creating a machine-readable trust layer that does not depend on behavioral reputation.

This paper emerged from a series of working conversations in March 2026 and is intended for collaborative development. The framework, risk model, and certification schema are proposals for discussion. Comments, critiques, and extensions are welcome.

References

Identity & Discovery

- Visa, "Visa Introduces Trusted Agent Protocol: An Ecosystem-Led Framework for AI Commerce," October 14, 2025. https://corporate.visa.com/en/sites/visa-perspectives/newsroom/visa-unveils-trusted-agent-protocol-for-ai-commerce.html
- ERC-8004, "Trustless Agents," Ethereum Improvement Proposal. https://ethereum-magicians.org/t/erc-8004-trustless-agents/25098
- Skyfire, "Know Your Agent (KYA)." https://docs.skyfire.xyz/docs/introduction-1

Wallets & Permissions

- MoonPay, "MoonPay Launches the Open Wallet Standard," March 23, 2026. https://www.moonpay.com/newsroom/open-wallet-standard
- CoinFello / OpenClaw, "OpenClaw Skill for AI Agent Transactions," March 2026. ERC-4337 smart accounts with ERC-7710 delegations.

Payment Protocols

- Coinbase, x402 protocol v2, December 18, 2025. https://docs.cdp.coinbase.com/get-

started/changelog

- Stripe / Tempo, “Machine Payments Protocol (MPP),” March 18, 2026. https://stripe.com/blog/machine-payments-protocol
- Tempo, “Tempo Mainnet is live,” March 18, 2026. https://tempo.xyz/blog/mainnet/

Governance & Standards

- NIST / CAISI, “Announcing the AI Agent Standards Initiative: Interoperable and Secure,” February 17, 2026. https://www.nist.gov/news-events/news/2026/02/announcing-ai-agent-standards-initiative-interoperable-and-secure
- Cloud Security Alliance, CSAI Foundation launch, RSA Conference 2026.

Legal

- Directive (EU) 2024/2853 of the European Parliament and of the Council (Revised Product Liability Directive). Entered into force December 8, 2024; member state transposition deadline December 9, 2026. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32024L2853
- EUR-Lex summary, “Defective products — liability.” https://eur-lex.europa.eu/EN/legal-content/summary/defective-products-liability.html

Agent Memory & Conditioning

- OpenClaw, “System Prompt” documentation. https://docs.openclaw.ai/concepts/system-prompt
- OpenClaw, “Context” documentation. https://docs.openclaw.ai/concepts/context
- OpenClaw GitHub Issue #9386, “[Feature]: Working Memory for Long-Horizon Agent Tasks.” https://github.com/openclaw/openclaw/issues/9386

Research

- “Autonomous Agents on Blockchains: Standards, Execution Models, and Trust Boundaries,” arXiv, late 2025. https://arxiv.org/html/2601.04583v1
- “AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents,” ICSE 2026. https://cposkitt.github.io/files/publications/agentspec_llm_enforcement_icse26.pdf
- “TRiSM for Agentic AI: A Review of Trust, Risk, and Security Management in LLM-based Agentic Multi-Agent Systems,” ScienceDirect, 2025. https://www.sciencedirect.com/science/article/pii/S2666651026000069