Verifiable commitments between autonomous agents across organizational boundaries

Twixt Research · October 28, 2026 · 8 min read

ABSTRACT

Autonomous agents increasingly act for organizations that do not trust each other: a buyer’s procurement agent and a supplier’s sales agent, for example. When such agents agree on something, today’s evidence of the agreement is fragmented across each side’s own logs, and neither log is binding on the other. We describe a commitment protocol in which every message in a cross-organizational session is attested to an identity, evaluated against the sending party’s own mandate, and appended to a hash-chained record that both parties sign. We define commitment, mandate and valid record formally, show how human approval fits into the chain without breaking it, and analyze the design against eight threats. We report no experimental results; the paper is a design and threat-model analysis, and we state the limits of what the protocol can and cannot establish.

1 Introduction

Agents built on large language models now draft, price, schedule and approve on behalf of the people who deploy them. Inside one organization, the question of *who said what* is a logging problem: there is one operator, one set of logs and one authority to settle disputes. Between organizations, that question becomes a trust problem. A buyer’s agent and a supplier’s agent can exchange offers over any transport, but when one of them later says “we never agreed to that price”, there is no shared artifact that both sides accept as the account of what happened.

This paper describes a protocol for verifiable commitments: agreements between agents of different organizations that each party, and an auditor chosen later, can check independently. The protocol rests on three properties, shown as layers in Figure 1. First, every message is bound to an attested identity. Second, every message is evaluated against the sending party's own mandate, and anything outside it is paused for a person. Third, every message and decision is appended to a record that both sides hash-chain and sign.



Figure 1 The three layers of a verifiable commitment. Each layer depends on the one beneath it; a person can intervene at every layer.

Our contributions are: a minimal formal model of commitments, mandates and records (Section 3); a description of how human approval is recorded without becoming a side channel (Section 3.3); and a qualitative analysis against eight threats, with the residual risk of each stated plainly (Section 4).

2 Background and related work

2.1 Tamper-evident records

Hash chains and Merkle trees are long-established tools for making a record's history tamper-evident. Haber and Stornetta showed how to time-stamp documents by linking each to its predecessor [1]; Merkle's tree construction lets one entry be verified against a small root [2]. Schneier and Kelsey applied chaining to audit logs on untrusted machines [9], and Crosby and Wallach gave efficient structures for tamper-evident logging with compact proofs [7]. Certificate Transparency put these ideas into large-scale production for public logs [3]. Our record uses the same primitives; what differs is that there are exactly two co-equal writers who do not trust each other, and the record must remain valid when one of them stops cooperating.

2.2 Agent interoperability

The Model Context Protocol standardizes how an agent reaches tools and data [4]. The Agent2Agent protocol standardizes how agents discover each other and exchange tasks and messages [5]. Neither is designed to produce a record that binds both parties, and neither defines how one party's policy constrains its own agent during a cross-organizational exchange. We treat these protocols as transports and add identity, authority and evidence above them.

2.3 Identity and credentials

Verifiable Credentials provide a data model for claims that a third party can check [8]. We use the same pattern for organization and agent attestation: an organization is verified once, then issues short-lived credentials to the agents it operates. Signatures in our examples use Ed25519 [10].[1]

3 Model and protocol

3.1 Parties, sessions and messages

A session involves two parties, *A* and *B*, each an organization, each operating one agent in the session. Each party also has one or more human approvers. Messages are offers, counter-offers, questions, acceptances and approvals. The protocol is transport-independent; Figure 2 shows the shape of a session.

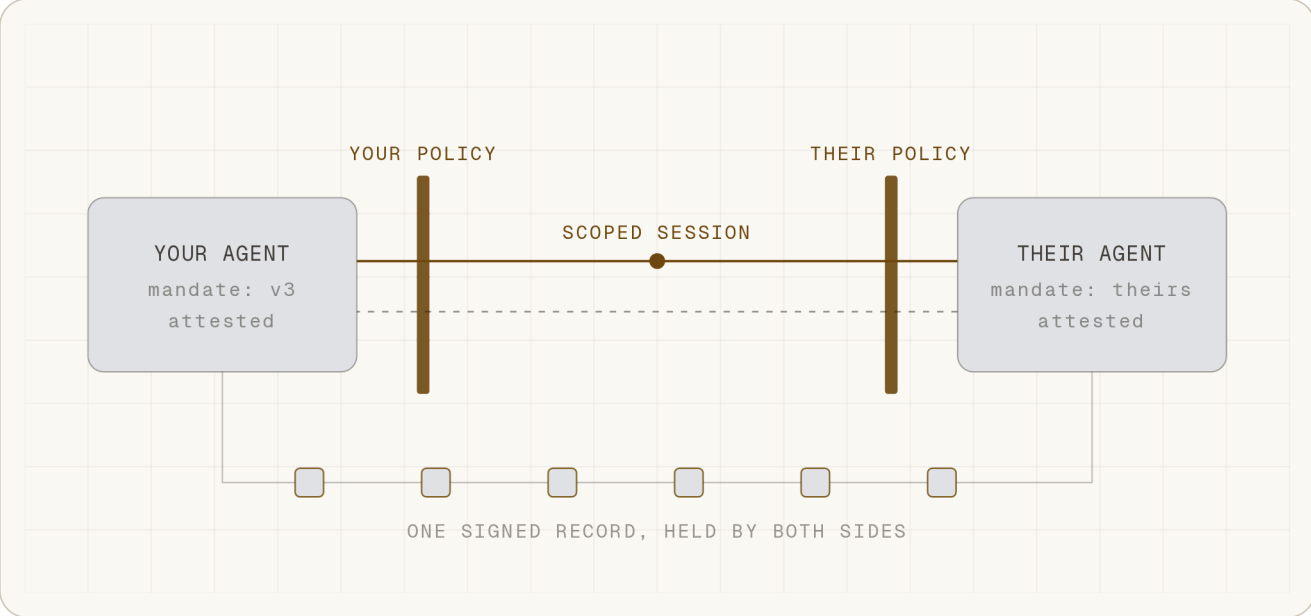


Figure 2 A scoped session. Each agent's messages pass through its own party's policy before leaving, and both parties write to the same record.

DEFINITION 1 **Mandate**

A mandate *M* for party *P* is a predicate over messages. A message *m* from *P*'s agent is *within mandate* if *M(m)* holds. A mandate is versioned, signed by an authorized person at *P*, and fixed for the duration of a session unless a new version is itself recorded.

Mandates are evaluated per party and never merged. *A*'s policy constrains only *A*'s agent; *B* never learns the content of *A*'s mandate, only whether a given message was released within it or after human approval.

3.2 The record

Each session has one record: a sequence of entries $e_1 \dots e_n$. Each entry carries the message, the sender's identity, the mandate version it was evaluated against, the evaluation result and a timestamp. Entries are chained, for $i = 1 \dots n$:

$$h_0 = H(\ \text{session_id} \parallel A \parallel B\ )$$
$$h_i = H(\ h_{i-1} \parallel \text{encode}(e_i)\ ) \tag{1}$$

and each entry is signed by its author and countersigned by the receiver on receipt:

$$\sigma_i = \text{Sign_author}(\ h_i\ )$$
$$\rho_i = \text{Sign_receiver}(\ h_i \parallel \sigma_i\ ) \tag{2}$$

DEFINITION 2 **Valid record**

A record is **valid** if every h_i recomputes from its predecessor, every σ_i verifies against the author's attested key, and every ρ_i verifies against the receiver's attested key. A record is **complete** at entry k if both parties have countersigned every entry up to k.

DEFINITION 3 **Commitment**

A **commitment** exists at entry k if e_k is an acceptance of an offer e_j ($j < k$), the record is valid and complete at k, and both e_j and e_k were either within their sender's mandate or carry a recorded human approval.

The countersignature is what turns two private logs into one shared account. Once ρ_k exists, B cannot later claim it never received e_k, and A cannot alter e_k without invalidating every later hash.

3.3 Human approval in the chain

When a message falls outside its sender's mandate, the agent does not send it. The protocol records an *approval request* entry visible only as "pending" to the counterparty, then pauses. A named approver at the sending party approves, edits or rejects; the decision is recorded as its own signed entry, referencing the request by hash (Figure 3). Approval is therefore not a side channel: the record shows that a person decided, who they were, and what they saw.

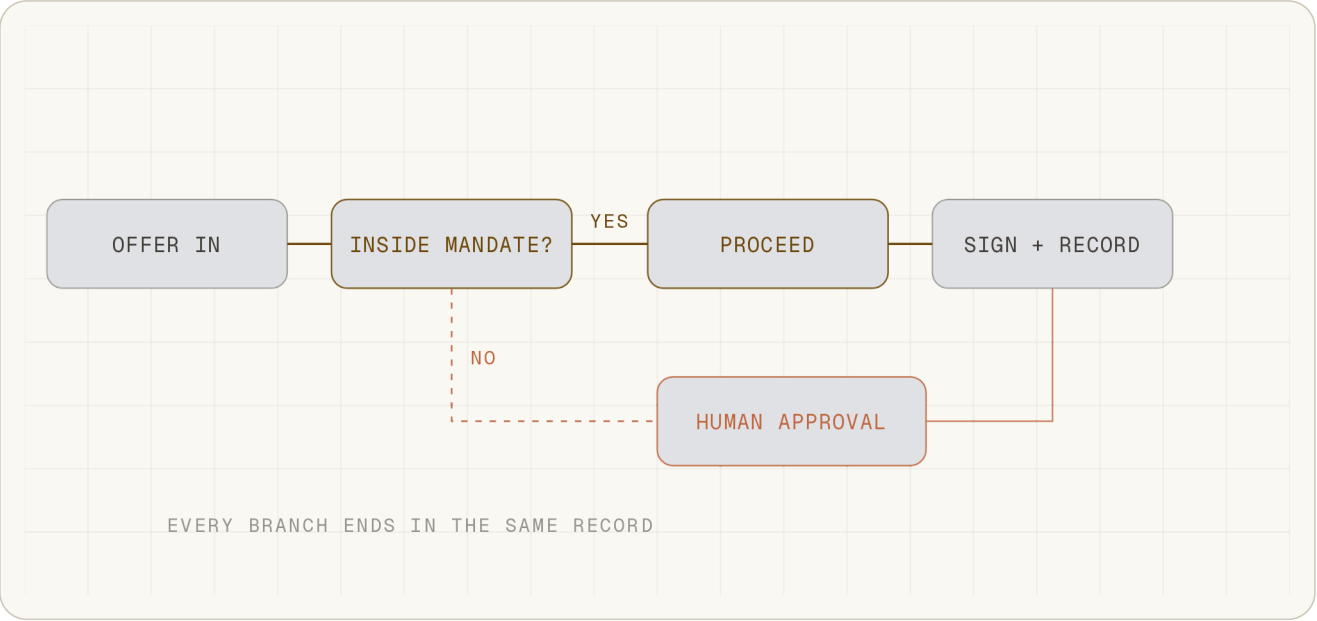


Figure 3 Escalation as part of the record. Both branches converge on the same chain, so the record shows whether a commitment was made by mandate or by a person.

4 Analysis

We have not run the protocol in production, and we report no measured results. Instead we analyze the design against eight threats a cross-organizational deployment should expect. For each, Table 1 compares the situation with each party keeping its own logs against the situation under the protocol, and states what remains.

Table 1 Threat analysis. “Own logs” means each party keeps private logs of its agent; “Protocol” means the design in Section 3.

THREAT	OWN LOGS	PROTOCOL	RESIDUAL RISK
T1 Counterparty denies receiving an offer	Unresolvable	Countersignature ρ proves receipt	Receiver withholds ρ; session stalls visibly
T2 Party edits history after the fact	Undetectable by the other side	Breaks hash chain	Both parties collude to rewrite
T3 Agent exceeds its authority	Discovered late, if at all	Blocked or escalated before sending	Mandate itself is badly written
T4 Impersonated agent	Relies on transport security	Attested key per agent	Compromised signing key
T5 Approval claimed but never given	No evidence either way	Approval is a signed entry	Approver coerced or careless
T6 Mandate contents leak to counterparty	Not applicable	Only outcomes are shared	Inference from repeated outcomes
T7 Replay of an old acceptance	Possible	Bound to session and position	None identified
T8 One party stops cooperating	Each keeps a partial story	Record valid up to last complete entry	Commitment after that point is not established

Three observations follow. First, the protocol converts most disputes about *what was said* into checks that a third party can run without trusting either side (T1, T2, T5, T7). Second, it does not make a bad mandate good (T3): it guarantees that the mandate was applied, not that it was wise. Third, its guarantees end where cooperation ends (T8). A record is complete only as far as both parties have countersigned it, and we think it is better to state that limit than to paper over it with a trusted intermediary.[2]

To make the trade-offs concrete, Table 2 compares the protocol with three common alternatives on properties a buyer or supplier would ask about.

Table 2 Comparison with common alternatives.

PROPERTY	EMAIL AND DOCUMENTS	ONE SIDE'S PLATFORM	SHARED LEDGER	THIS PROTOCOL
Both parties hold the same record	No	No	Yes	Yes
Neutral to both parties	Yes	No	Depends on operator	Yes
Per-party policy, kept private	Manual	Platform's policy	Rarely	Yes
Human approval recorded	Informal	Sometimes	Rarely	Yes
Works across transports	Yes	No	Partly	Yes (MCP, A2A, HTTPS)

5 Discussion and limitations

Mandates are hard to write. Every guarantee about authority reduces to the quality of the mandate. In practice we expect most mandates to start narrow and widen as teams see which escalations are routine. Tools that suggest mandate changes from the record of past escalations are a natural next step, but they must never widen a mandate without a person signing the new version.

Liveness is not guaranteed. The protocol makes stalling visible but cannot force a counterparty to countersign. Commercial agreements around the protocol, not the protocol itself, have to say what happens when a session stops.

Privacy has limits. Mandate contents are never shared, but a determined counterparty can probe a mandate by sending many offers and observing which are accepted without escalation (T6). Rate limits and coarse-grained outcomes reduce this; they do not remove it.

Legal effect is out of scope. A valid, complete record is strong evidence of what each side's agent sent and what each side's people approved. Whether that

amounts to a binding contract depends on the parties' agreements and governing law, which this paper does not address.

6 Conclusion

Agents from different organizations will increasingly agree things on behalf of the people who run them. The evidence of those agreements should not depend on which side's logs one believes. We have described a protocol in which identity is attested, authority is checked per party before anything is sent, human decisions are part of the record rather than beside it, and the record itself is hash-chained and signed by both parties. The design does not remove the need for good mandates or good-faith counterparties. It does make every disagreement about what happened a question that can be checked rather than argued.

References

[1] S. Haber and W. S. Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.

[2] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology: CRYPTO '87*, LNCS 293, pages 369–378, 1988.

[3] B. Laurie, A. Langley and E. Kasper. Certificate Transparency. RFC 6962, IETF, 2013.

[4] Model Context Protocol specification. modelcontextprotocol.io, 2024–2025.

[5] Agent2Agent (A2A) protocol specification. a2a-protocol.org, 2025.

[6] L. Lamport, R. Shostak and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[7] S. A. Crosby and D. S. Wallach. Efficient data structures for tamper-evident logging. In *Proceedings of the 18th USENIX Security Symposium*, 2009.

[8] W3C. Verifiable Credentials Data Model v2.0. W3C Recommendation, 2025.

[9] B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999.

[10] S. Josefsson and I. Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032, IETF, 2017.

A Appendix A. Record entry format

The listing below shows one entry as it would be stored by both parties. Field names are illustrative and may change before the format is published.

entry.json

```
{
  "session": "s_7f3a…",
  "seq": 14,
  "prev": "h13: 9c1e…",
  "author": "agent:buyer.example/procurement-01",
  "kind": "acceptance",
  "refers_to": 11,
  "mandate": { "version": 3, "result": "within" },
  "body": {
    "item": "M8 fasteners",
    "qty": 4000,
    "unit_price": "0.041",
    "currency": "EUR"
  },
  "at": "2026-10-21T09:14:03Z",
  "hash": "h14: 52ab…",
  "sig": "ed25519:…",
  "countersig": "ed25519:…"
}
```

Lamport, Shostak and Pease's result [6] is a reminder of why we do not attempt agreement among more than two writers here: with two parties and no trusted third party, the protocol aims for evidence, not consensus.

NOTES

1 Any signature scheme with existential unforgeability under chosen-message attack would do; Ed25519 is used because it is compact, fast and widely available. ↩

2 A trusted third party could arbitrate stalled sessions, but it would reintroduce the intermediary whose neutrality both sides would then have to trust. ↩