

Confidential Pulse: Authorisation-Preserving Conversational Access to Federated Confidential Government Records

Khaled Almi'ani

`kalmiani@hct.ac.ae`

Higher Colleges of Technology

Camille Zufferey

Finswiss Research

Salil S. Kanhere

UNSW Sydney

Nur Siyam

Higher Colleges of Technology

Research Article

Keywords: Confidential data sharing, Permissioned blockchain, Hyperledger Fabric, Authorisation preservation, Grounded generation, Multi-agent systems

Posted Date: October 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11144191/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Confidential Pulse: Authorisation-Preserving Conversational Access to Federated Confidential Government Records

Khaled Almi'ani^{1*}, Camille Zufferey², Salil S. Kanhere³, Nur Siyam¹

^{1*}Department of Computer Information Science, Higher Colleges of Technology, United Arab Emirates.

²Finswiss Research, Sydney, NSW 2100, Australia.

³The University of New South Wales, Sydney, Australia.

*Corresponding author(s). E-mail(s): kalmiani@hct.ac.ae;

Contributing authors: camille@finswiss.com.au; salil.kanhere@unsw.edu.au; nsiyam@hct.ac.ae;

Abstract

Government records that cannot be published must still move between organisations. Data lakes concentrate them and point-to-point integration records nothing centrally. Confidential Pulse makes conversational access to those records bounded rather than trusted. In reconstruction mode, executions with the same query, plan and deployment state give the same observable response when their authorised records agree. The principal is bound once. Authorisation is re-derived at release from ledger state, and content is reconstructed from the authorised payload rather than filtered after a model writes it. Hyperledger Fabric channels and private data collections enforce need-to-know in the protocol, while anchors keep records tamper-evident. Across 20,000 randomised trials, reconstruction mode yields no out-of-scope disclosure. Prose written by the model leaks in 2,068 of them. We measure the system live across five cloud regions. Three channels stay outside the invariants: the cross-domain join, degraded-answer differencing, and what an anchor discloses to channel members.

Keywords: Confidential data sharing, Permissioned blockchain, Hyperledger Fabric, Authorisation preservation, Grounded generation, Multi-agent systems

1 Introduction

A modern government runs on data that crosses organisational lines. A single citizen interaction can touch records for land and tax as well as civil status, licensing, and welfare. Those records sit in different ministries, agencies, and municipalities. Each body has its own mandate and its own duty of confidentiality. Two trends pull on the same data at once. The first is open government data, where an organisation publishes

non-sensitive records on a public portal for anyone to reuse [1]. The second is integration, where joined-up services require sensitive records to move between organisations rather than out to the public. Open government data cannot serve the second need. The data that must move between organisations is exactly the data that must not become public.

How can government organisations share sensitive records with one another on a strict need-to-know basis, so that the result is confidential

and tamper-evident and independently verifiable? Two standard answers each solve part of the problem. The first is a central data lake. It copies every organisation’s records into one warehouse that is easy to query. That same warehouse becomes a single high-value target, and it asks every participant to trust one operator. The second answer is a mesh of point-to-point integrations. This approach keeps data in place but multiplies bilateral connectors. The count of connectors grows as $O(n^2)$ in the number of organisations. No shared tamper-evident record of who shared what survives, so no participant gains a view of the whole.

Confidential Pulse adopts a different foundation, shown in Figure 1. Each organisation is a member of a permissioned ledger [2], which turns sharing into a recorded and first-class act. The ledger commits a tamper-evident anchor on-chain. The sensitive payload itself reaches only entitled members through Fabric channels and private data collections [3]. Need-to-know is therefore enforced at the protocol level rather than by policy alone. A hierarchy of AI agents sits above the ledger and renders the dispersed records answerable. A leaf agent reads one organisation’s records, and the domain and sector agents above it aggregate those reports upward. A master agent at the apex presents a large language model chat interface. Through it an authorised official poses a single plain-language question across government.

Coupling a language model to confidential data introduces two risks. Leakage discloses a record to an official who is not entitled to it. Fabrication states a claim that no record supports [47]. Confidential Pulse constrains both risks through two runtime-enforced invariants. Authorisation preservation requires every released item to be one that the asker’s organisation may read and whose clearance requirement the asker personally holds. Grounding requires every released sentence that asserts something about a record to resolve to a confirmed on-chain record. It builds on retrieval-augmented generation [17] and on entailment checking [41, 45], and it runs as a gate rather than as a retrospective score.

A whole-of-government interface also has to be monitored. We therefore describe an oversight plane on the same hierarchy that consumes metadata-only audit signals in Section 8. The confidentiality of the plane is argued structurally,

whereas its detection accuracy is left to future evaluation.

Three research questions organise the investigation. The first asks how to share sensitive records across organisations so that the result is confidential and tamper-evident and independently verifiable. The second asks how one conversational interface can answer questions spanning many organisations without ever exceeding the asker’s authorisation or fabricating an unsupported claim, even when the language model is untrusted or adversarially manipulated. The third asks what governance and identity and security guarantees a deployment must provide, and which risks remain after them.

This paper makes four contributions. The first is the security property itself, stated end to end rather than as a list of mechanisms. The property takes a non-interference form. Fix the query and its clamped plan. Fix the public configuration and the availability state as well. Two executions in reconstruction mode then return the same observable response whenever the records authorised for the bound principal agree. Three things establish it. The principal is bound once at the master and propagated unchanged, so no descendant can enlarge it. The release gate re-derives entitlement from ledger state rather than trusting agent-declared metadata, so a compromised or prompt-injected agent cannot pass an item by mislabelling it. And the released text is rebuilt from the authorised payload under (2) rather than inspected after a model writes it, so no model behaviour can put a value into an answer that the cited record does not hold. We state the property, pin it to the predicate the implementation enforces, and then attempt to falsify it at scale.

The second is the architecture and its Fabric realisation: organisations map to members, need-to-know maps to channels and private data collections, and an identity model binds an authenticated official to Fabric credentials and attribute-based clearances with a precise account of revocation and issuer-key rotation. The agent hierarchy sits above this. It is a scalability and routing tool, and its subtrees are the boundaries at which isolation contains a failure. Its benefits grow with the number of organisations. It is not part of the security argument, which rests on the ledger and the gate.

The third is the evaluation. We separate the two release modes and recommend one. Reconstruction mode admits no out-of-scope disclosure across 20,000 adversarial trials. Prose mode filters rather than rebuilds. It admits a disclosure in 2,068 of the same 20,000 trials, carrying withheld values in 2,394 released sentences. The residual concentrates in comparative assertions. It survives complete notation coverage. A value-matching content clause either rejects 50% of true threshold answers or admits 100% of false ones. It never evaluates the comparison. We add a federated non-blockchain baseline of signed records over access-controlled stores. It matches the ledger on confidentiality: 0 hostile admissions in 115 probes, as the ledger also has 0. It separates elsewhere. It detects 0 of 20 unilateral post-hoc rewrites where the ledger detects 20. It offers no independently verifiable history. We measure the deployment with one operating-system process per organisation communicating over sockets, and implement the concurrent sibling fan-out the architecture implies. That change cuts the growth of query latency with link delay from 29.7 to 16.4 milliseconds per millisecond. The released answer stays byte-identical.

The fourth is an account of what the invariants do not bound, which we treat as a result rather than a caveat. Equation (1) bounds the records an answer contains and not what an asker derives by combining them, and the master performs exactly such a combination. On a synthetic population of 10,000 subjects, each domain alone leaves no subject uniquely identified, while the three-way join the master performs singles out 3.31% of them. Separately, degrading a chosen dimension changes the join’s answer set. The difference is computed from authorised records and reveals partial-join results the question did not ask for. We give the fail-closed join that removes it. Third, an anchor discloses to every channel member what each organisation holds and withholds, and for the Land peer 71% of the anchors it can see stand for payloads it cannot read. The commitment inside that anchor is the one implementation defect among the three. In the evaluated implementation it is an unsalted digest of a structured payload. A channel member not entitled to the payload recovered 10 of 10 low-entropy records in

1.9 seconds. The corrected design salts the commitment, and the same attack on a salted ledger recovers 0 of them.

The remainder of the paper is organised as follows. Section 2 reviews related work and positions the system. Section 3 sets out the operational context and the actors. Section 4 presents the architecture and Section 5 its Fabric realisation. Section 6 covers identity and revocation, Section 7 the agent protocol and its invariants, and Section 8 the audit-signal plane. Section 9 gives the threat model and the security properties. Section 10 draws the concepts together through a worked example. Section 11 reports the evaluation, Section 12 the risk register, and Section 13 governance and regulatory compliance. Section 14 concludes.

2 Related Work and Positioning

Confidential Pulse sits at the intersection of four literatures, and this section reviews each in turn before summarising the positioning in Table 1.

Open government data studies the publication of public-sector data for reuse [1]. Its premise is that the data is publishable, which is exactly what fails for the cross-organisation flows we target. Permissioned ledgers such as Hyperledger Fabric [2] replace open consensus with known and accountable organisations. Fabric adds per-channel ledgers and private data collections that keep payloads off the shared ledger while anchoring their hashes [3]. A critical assessment of blockchain in government cautions that its frequently claimed benefits are overstated and remain conditional on sound governance [5]. Public-registry work has examined the same anchoring pattern, though as design analysis rather than as deployment. An assessment against records-management standards finds that a proposed blockchain land registry strengthens integrity without guaranteeing that the record was reliable when created [6]. A blockchain design for land titling in India remains at the architecture stage [7]. Attribute-based access-control schemes on Fabric enforce fine-grained access to stored data [8]. Federated data-space infrastructures such as International Data Spaces and GAIA-X govern

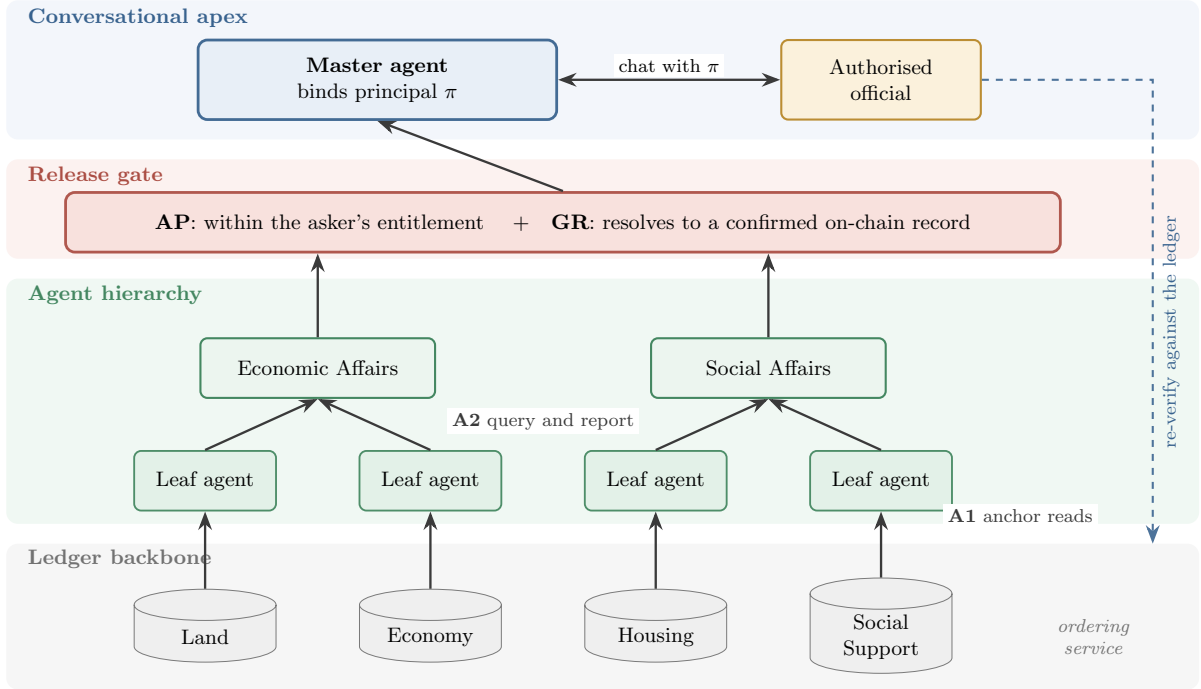


Fig. 1 Confidential Pulse architecture. Queries route down and data flows up, while authority never flows down.

connectors and usage contracts across organisations [9]. Confidential Pulse builds on this foundation and extends enforcement from stored data to generated answers, and it adds a conversational tier that none of these systems addresses.

Self-sovereign identity is a design philosophy in which individuals hold and control their own identity data rather than depending on centralised providers [10, 11]. Two W3C standards realise this model [11]. Verifiable credentials rest on a trust triangle of issuer, holder and verifier [11]. An issuer attests to attributes of a holder. The holder presents credentials to a verifier, who checks them before accepting them [12]. Decentralized Identifiers give these parties globally unique identifiers [11, 12]. Selective disclosure lets the holder reveal only chosen attributes. The European Union adopted this model in Regulation (EU) 2024/1183, known as eIDAS 2.0 [12, 13]. Every Member State must offer an EU Digital Identity Wallet [14]. The Architecture and Reference Framework gives the technical specifications of the wallet [13]. Citizens, residents and businesses can store attested attributes in the wallet [13]. The United States analogue in NIST SP 800-63-4 [15] likewise recognises a subscriber-controlled wallet model.

The distinction we draw is structural. Self-sovereign identity and verifiable credentials solve identity and attribute attestation. Their unit of exchange is a credential about the holder, presented by that holder to a relying party at a moment of interaction. The trust question is whether a claim about a subject is authentic and unrevoked. Confidential Pulse solves a different problem, namely inter-organisational sharing of record payloads with a shared audit trail and conversational query across holdings. Here the data concerns many subjects at once, and the parties exchanging it are organisations rather than the individuals described. The governance need is a permissioned and append-only ledger of who accessed what. A holder-centric presentation flow does not express bulk custodial exchange between institutions, and it provides no durable cross-organisational audit substrate. A Fabric audit ledger in turn gives citizens no holder-controlled selective disclosure. The two are complementary layers rather than substitutes. A language model in the loop does not reach records on its own authority. The master agent holds no standing access, and every answer is bounded by the asker's

own credential as Section 6 describes. Where citizen consent is the governing basis for a sharing relationship, a wallet layer is the right complement to gate the issuance of records into a collection. We make that integration explicit as future work rather than subsume it. Recent work equipping language-model agents with decentralized identifiers and verifiable credentials concerns agent authentication and delegated authority, and it reports limitations once the model alone controls the security procedures [16]. That finding reinforces our choice to keep authorisation on the platform rather than in the model.

Retrieval-augmented generation grounds a model in retrieved context [17], but the retrieved context can leak. RAG pipelines are vulnerable to prompt-driven extraction of their private retrieval corpus [18]. A proof of concept couples document-level access control with RAG for sensitive patient records, using a document-level access model in a single application [19]. Role-based access control supplies the policy vocabulary that richer schemes build on [20]. Attribute-based schemes carry that vocabulary onto Fabric [8], and cryptographic constructions secure the store itself [21]. A separate line of work concerns attacks on the model. Instructions hidden in retrieved content can hijack a language-model application through indirect prompt injection [22]. The attack surface has been formalised and benchmarked [23]. Secure-by-design patterns confine what an untrusted model can cause. The dual-model pattern keeps privileged control away from untrusted text by passing that text as a symbolic variable [25]. CaMeL extracts control and data flow from a trusted query and enforces capability policies at tool calls [24]. A catalogue of six such patterns names the Action-Selector, in which the model chooses among predefined calls and never observes their results [25]. Programmable privilege control enforces symbolic rules on tool invocations outside the model loop [26]. All of this descends from the confinement problem [27].

The master’s parse follows the Action-Selector pattern, because the model picks a plan from a predefined vocabulary that the clamp enforces. We claim that use as a retargeting rather than as an invention. In reconstruction mode the model chooses nothing on the answer channel, and the released text is rebuilt from authorised records.

Two differences from that literature are substantive. That literature governs flow outward to a third party and inward to the control path, whereas the property here governs flow to the asker, and those quantify over different populations. That literature also trusts the person asking and distrusts the data, whereas the agent hierarchy here legitimately reads more than any single asker may see, which is the asymmetry that creates the problem. Enforcement therefore sits at the user-facing answer, which is the boundary those designs leave open, and the released set is itself the output of an authorisation decision re-derived at release time.

Answering from the authoritative store under the asker’s own rights, rather than filtering results afterwards, is much older than any of this. Query modification appends the authorisation predicate to the user’s query before execution [28], a technique productised as label-based and policy-function access control. Fine-grained access control through query rewriting names two regimes [29]. The Truman model answers every query against a personal view of the database, which can silently mislead. The non-Truman model answers exactly or rejects. Later work formalised when a query is authorised under authorisation views [30] and proved that no computable algorithm attains security and soundness and maximality together over the full relational calculus [31]. The same work shows the existential fragment decidable, and optimal algorithms do exist there.

Re-derivation is that idea, and we do not claim the mechanism as new. What is new is the argument that it is forced. Query modification was a convenience in a setting where the answer channel was the query result itself. Once a generative intermediary composes prose beside the result, filtering that intermediary is unsound. Notation has no finite enumeration. Re-derivation stops being one option among several. Two further parts have no analogue in that literature. The construction recurses through aggregates, so every field of a joined row derives from re-derived parts. The re-read also recomputes the payload digest against an owner-endorsed anchor, which detects a store that diverged from the endorsed record. We adopt the Truman regime deliberately, answering within entitlement rather than refusing when an exact answer is unavailable, and the citation set

attached to every answer makes the extent of that view visible to the asker.

Confidential Pulse also treats every retrieved record and every model output as untrusted, and Section 11 drives a randomised adversarial renderer and parser against it rather than only benign inputs. The gate holds in reconstruction mode and carries a measured residual in prose mode.

Whether generated text is supported by its sources is studied under attribution and faithfulness. The Attributable to Identified Sources framework defines attribution as a human-evaluation standard [39]. ALCE benchmarks automatic citation precision and recall [40]. TRUE standardises factual-consistency meta-evaluation across eleven datasets and finds that large-scale natural language inference and question generation with answering achieve strong and complementary results [41]. FActScore decomposes output into atomic facts verified individually against sources [42]. All four measure attribution after the fact. Confidential Pulse instead enforces it. Every sentence must resolve to a confirmed on-chain citation before release, and a sentence that fails falls through to a deterministic template that is grounded by construction. Our gate uses an entailment model [45]. That is one of the two metric families TRUE found strongest, applied as a release condition rather than as a score.

The mechanisms we use are individually established. Fabric channels and private data collections are long established, as are hash anchoring and hierarchical access control. Retrieval-augmented generation and entailment-based grounding and agent orchestration all predate this work. Prior language-model agent work is orthogonal to authorisation. Multi-agent frameworks coordinate through a shared conversation or message pool with no per-agent data scope [48, 49]. Single-agent methods for interleaved reasoning and acting [50] and for verbal self-reflection [51] assume unrestricted use of whatever tools and context they are given. None of them models an entitlement that a generated answer must not exceed. The composition is the contribution, and it is delimited as follows. The authorisation decision that bounds a generated answer is re-derived at release time from an authority that is not a party to the generation, that no single participating organisation can unilaterally alter, and against which a store that has

diverged from the endorsed record is detectable. The re-derivation rebuilds the answer text rather than inspecting it.

Each clause answers a specific alternative. Rebuilding rather than inspecting is forced by the notation argument. An authority that is not a party rules out a capability store held by whichever host runs the agent. An authority no single member can alter rules out one organisation’s own authorisation view, since in a federation that organisation is a party to the dispute. Detectable divergence is the clause that requires a ledger rather than any replicated store, because a system that is simultaneously the authority and the store returns a self-consistent answer over corrupted data, and the endorsed digest is what makes that visible. Cross-organisational schemes on this platform have addressed role-based access across jurisdictions [37] and confidential sharing through channels rather than private data collections [38]. We are not aware of a peer-reviewed evaluation of private data collections as the need-to-know construction in a public-sector cross-agency setting.

3 Operational Context and Actors

A federation of government organisations operates the system under a standing data-governance board. That board has representation from each member and from the data-protection authority. No single organisation owns the platform. Each member runs its own ledger nodes and its own leaf agent.

Five kinds of actor appear in the rest of the paper. A records officer in an owning organisation shares a record by invoking the **ShareRecord** transaction. That officer chooses the collection and therefore the need-to-know set, under least-privilege defaults and board policy. An authorised official in a consuming organisation asks the master agent a plain-language question. Every answer that official receives is bounded by their credential and clearances, and every claim in it is cited. A divisional compliance officer authors the machine-readable policy schema at the division tier and adjudicates whatever the proposed audit plane flags. An auditor or data-protection officer replays the immutable log to see the lawful basis recorded

Table 1 Positioning against alternative architecture classes.

Approach	Cross-org sharing	Need-to- know conf.	Tamper- evidence	Unified Q&A	Per-claim attribution	Untrusted model
Open data portal	–	–	(~)	–	–	–
Central data lake	✓	(~)	–	(~)	–	–
Point-to-point / MoUs	✓	(~)	–	–	–	–
SSI / VC (holder-centric)	(~)	✓	✓	–	–	–
Access-controlled RAG	(~)	✓	–	✓	(~)	(~)
Permissioned ledger alone	✓	✓	✓	–	–	–
Confidential Pulse	✓	✓	✓	✓	✓	✓

✓ satisfied, (~) partial, – not addressed. The comparison is qualitative and compares architecture classes rather than implementations. Per-claim attribution denotes the structural clause of (3) and not semantic faithfulness, which Section 7 bounds rather than guarantees.

for each share, although the log cannot confirm that the basis was legally valid. Replaying who read what needs the read-side log of Section 8, which is proposed rather than implemented. The governance board sets membership and channel and collection policy through recorded multi-party action.

The intended workload is low-throughput custodial sharing among a bounded and governed set of mutually distrustful but cooperating organisations. These are sharing events rather than high-frequency transactions. The system is not a citizen-facing identity wallet, which is the role of the self-sovereign identity layer discussed in Section 2. It is not a public data portal. It is also not a general assistant with standing access to government data. It is the substrate on which need-to-know sharing between organisations is recorded and made verifiable and made answerable under the asker’s own authority. We expect the same pattern to fit other federations of mutually distrustful but cooperating bodies, such as healthcare networks and regulator consortia. We have not evaluated it outside the synthetic government setting of this paper.

4 Reference Architecture

Confidential Pulse has three tiers, drawn in Figure 1. A ledger backbone makes sharing confidential and tamper-evident and auditable, an agent hierarchy interprets the dispersed records, and a conversational apex presents one chat surface. The rule that governs the whole stack is asymmetric. Data and verifiability flow up, while authority and write access never flow down. Agents read the ledger within a scope, and they neither write authoritative records nor widen who may read. Beside the data a second upward flow

would carry audit signals, each a piece of need-to-know-bounded metadata that the oversight plane of Section 8 would consume. A signal never carries a payload.

Six design goals drive the architecture. Confidentiality by construction is G1 and tamper-evidence with verifiability is G2. Auditable sharing is G3 and the absence of a central data warehouse is G4. Unified conversational access is G5, and least privilege end to end is G6. Goals G1 through G4 are properties of the ledger tier. G5 is the job of the agent tier. G6 is the cross-tier invariant, and it is the central design challenge. The ledger avoids a single warehouse of payloads, but the master agent and the on-premises model cluster do aggregate answers and audit metadata across organisations. They are themselves a valuable target.

The ledger backbone is a permissioned Fabric network. Each organisation is a Fabric Organisation with its own Membership Service Provider, certificate authority, and peers. Organisations that share form a channel, and each channel carries its own ledger. A private data collection inside a channel narrows disclosure of a payload to a named subset while the channel ledger holds only its hash [3]. The ledger is append-only and hash-linked.

The agent hierarchy mirrors the organisational tree. A leaf agent reads exactly one organisation’s scope and exposes summarise, detect, and answer skills. A non-leaf agent owns no ledger scope of its own. It decomposes a query to its children and aggregates their cited reports while preserving provenance and scope. The master agent at the top parses a natural-language question into a routing plan. It binds the official’s principal to the root query and renders the aggregated report. It holds no standing access, so with no principal

attached it can answer nothing. Because the chat interface is a capability, any non-leaf agent can be given it. A deployment therefore chooses how high to expose conversation, either as a whole-of-government master or as a scoped consultant over a single sector under the same invariants. That choice narrows what an answer can contain and never widens it.

The agents are not hosted services. They are subagents of an on-premises language model cluster inside the same trust boundary as the ledger. No record payload and no audit signal is ever sent to an external model. This is a hard requirement rather than an optimisation, and it bounds the model quality available to the grounding gate and to any future detector.

5 Ledger Realisation in Hyperledger Fabric

Fabric supplies the primitives that the design goals demand. Organisations act as members, which serves G1 and G4. Channels and private data collections are the unit of need-to-know, which serves G1 and G6. An immutable endorsed transaction log is the audit trail, which serves G2 and G3 [2]. A channel is the coarse unit, since only its member organisations hold its ledger at all. A collection is the fine unit. Fabric disseminates the payload only to the named Membership Service Providers and puts just its hash on the channel ledger. A collection definition names the entitled members, for example a subsidy-eligibility payload shared with the housing and welfare organisations only. Disclosure is therefore bounded by policy-as-code rather than by trust.

Every shared record is committed through chaincode in a fixed and organisation-neutral schema, so the agents above can read any organisation’s records uniformly. Figure 2 shows what that commitment writes and who can read each part, and Algorithm 1 gives the core **ShareRecord** transaction. In the corrected design it writes the private payload and a fresh 128-bit salt to the named collection with **PutPrivateData**. It writes the public anchor, which is the salted hash plus need-to-know metadata, to the channel ledger. The evaluated implementation omits the salt. Section 9 explains why the salt is needed, and EQ1 in Section 11 measures it on a salted ledger.

Algorithm 1: The corrected **ShareRecord** transaction, with salt.

```

Input: channel ch, collection col, record id r,
        payload p, owner o, signer s
if s ≠ o or o ∉ members(ch) or o ∉ members(col)
then
    | reject (endorsement or membership failure)
end
salt ← {0, 1}128
h ← SHA256(salt || canonical(p))
PutPrivateData(col, r, ⟨p, salt⟩) // payload and
    salt to named collection
a ← ⟨r, o, members(col), h, domain, ts⟩
PutState(“anchor~”+r, a) // public hash
SetEvent(“RecordShared”, a)

```

Every other result in this paper, including the live multi-region run, uses the evaluated implementation. The transaction then emits an event that leaf agents subscribe to. An endorsement policy requires the owning organisation’s signature, so a record cannot be shared in its name without its key. The anchor and the sharing event are immutable channel state. The question of who shared what with whom and when is therefore itself a ledger query, which is G3. A correction is a new **ShareRecord** that supersedes the old anchor, and the prior version remains in history. A record identifier belongs to the organisation that first published it. Requiring the owner’s signature proves only that a transaction was signed by whoever it names as owner, which any member can satisfy for any identifier. The chaincode therefore refuses an anchor whose identifier another organisation already holds. Without that refusal a member could publish over a neighbour’s identifier. The neighbour’s record would then vanish from every answer, on a chain that still verifies.

6 Identity, Credentials, and Revocation

Fabric channels and collections principally control membership at the level of the organisation, whereas the architecture also makes claims about individual officials and roles and clearances.

Each organisation runs its own Fabric certificate authority and Membership Service Provider, and enrolls its own officials. An official’s credential is a short-lived X.509 enrolment certificate whose extensions carry that official’s roles and clearance attributes. Authentication binds a principal $\pi = \langle \text{official}, \text{org}, \text{roles}, \text{clearances}, \text{serial} \rangle$ signed by

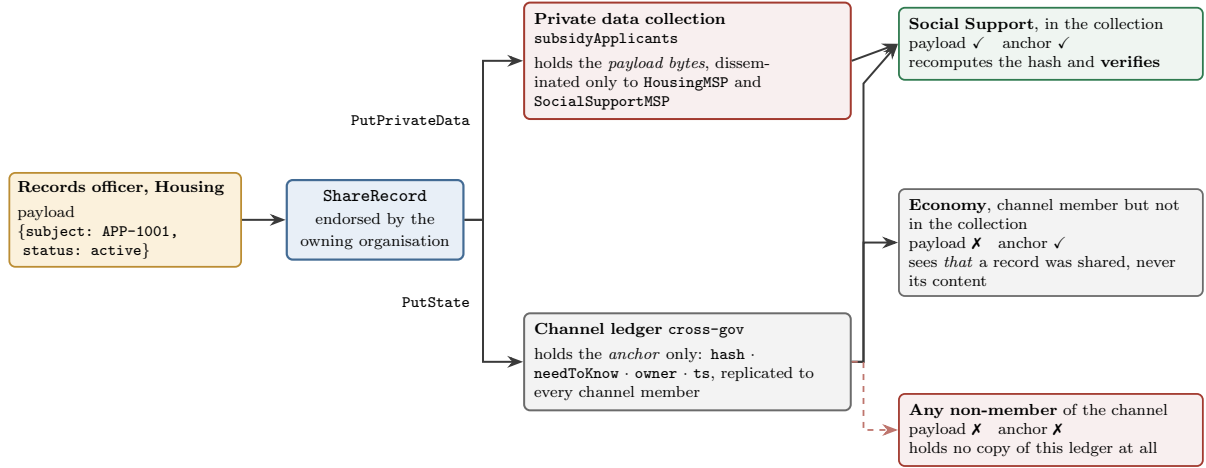


Fig. 2 One `ShareRecord` transaction, two writes, three different views.

the issuing authority. The master agent binds π to the root query only when the signature verifies under the current issuing key and the serial is absent from the revocation list. With no verifiable principal the master routes nothing.

Enforcement happens at two levels. Fabric membership enforces need-to-know at the level of the organisation, so a non-member peer never receives a channel’s payloads. An attribute check enforces clearance at the level of the individual. A clearance-gated collection such as property valuations requires the reading identity to present the `property-data` attribute, and the chaincode checks it rather than the client asserting it. The authorisation invariant of Section 7 then re-checks both conjuncts at release, re-derived from ledger state.

Removing a clearance from an official updates the organisation’s directory and revokes that official’s outstanding credentials by publishing their serials on the revocation list. A re-issued credential reflects the reduced attributes. Rotating the issuing key invalidates every credential signed under the old key and forces re-authentication. Offboarding revokes the official entirely.

A past answer was authorised by a credential that was valid at the time. Revoking that credential does not retract the answer, and it need not. Every released claim carries a citation that the recipient can re-verify directly against the ledger, independently of the credential, which is property P3 in Section 9. What revocation guarantees is forward-looking. The revoked credential

authorises no new answer, and the immutable log records both the original share and the revocation. An auditor can therefore reconstruct exactly what was authorised and when, once the read-side log of Section 8 is installed alongside it. Where a released answer must be treated as recalled, for instance after a clearance is revoked for cause, that is a governance action recorded on the ledger rather than a cryptographic undo.

7 Inter-Agent Protocol and Invariants

Every message carries the asker’s principal π , signed at authentication and immutable in transit. A leaf agent’s effective read set is exactly the records whose channel and collection it belongs to, which Fabric enforces. A query carries π together with an intent and a filter. A report is a list of cited items. Each item carries the on-chain key that supports it, the record’s need-to-know set `ntk`, and any clearance requirement `clr` inherited from the collection that holds it. An aggregate item’s citation is the union of its parts’ citations. Figure 3 gives the types and their binding to on-chain state. Algorithm 2 gives the master’s five orchestration steps. It authenticates and binds the principal. It parses the question to a plan and clamps that plan to known domains and intents. It routes to the relevant children and aggregates their cited reports. It gates each candidate against both invariants. It then renders only the survivors with their citations.

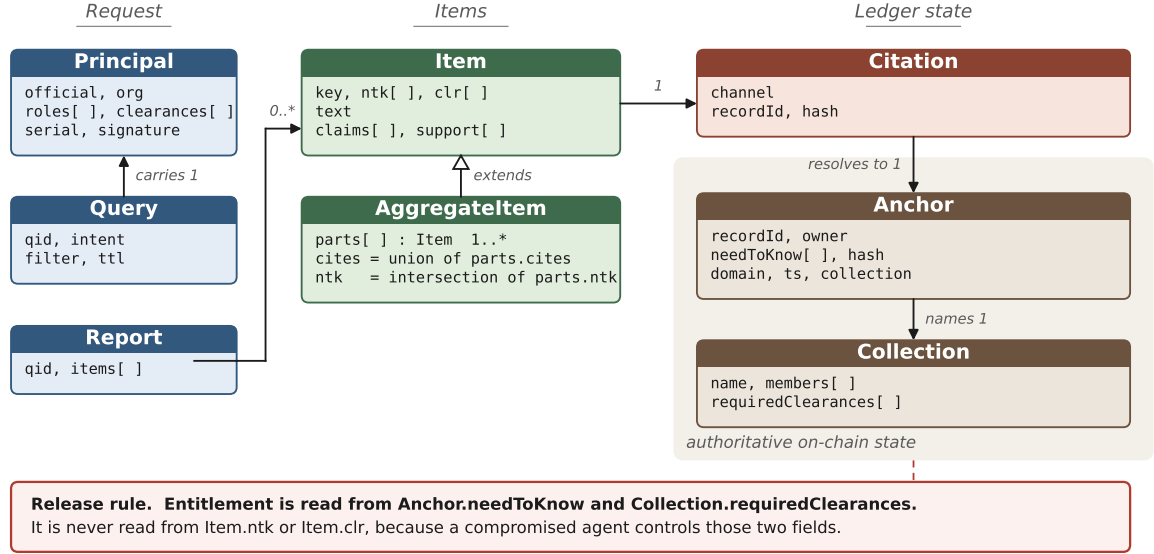


Fig. 3 Protocol data model and its binding to on-chain state.

Algorithm 2: Master-agent orchestration. The principal is bound once and propagated unchanged.

Input: question q , principal π , ledger L , auth service A

```

if  $\neg A.\text{valid}(\pi)$  then return "authentication failed"
plan  $\leftarrow \text{clamp}(\text{parse}(q))$  // unknown domains dropped
 $R \leftarrow \emptyset$ 
foreach child  $c$  routed by plan do
     $R \leftarrow R \cup c.\text{handle}(\text{descend}(\text{plan}, \pi))$  //  $\pi$  unchanged
end
 $R \leftarrow \text{aggregate}(R, \text{plan})$ 
 $R' \leftarrow \{x \in R \mid \text{anchoredAuth}(x, \pi, L) \wedge \text{grounded}(x, L)\}$ 
 $R^* \leftarrow \{\text{reground}(x, \pi, L) \mid x \in R'\}$  // text rebuilt from  $L$ 
 $C \leftarrow \text{render}(R^*)$  // untrusted model output
 $C^* \leftarrow \text{gateSentences}(C, R^*, L)$ 
foreach item in  $R^*$  not carried by  $C^*$  do append template
return  $C^*$  with the template fallback

```

The first invariant is authorisation preservation. Let π be the asker's principal and let R be the assembled report. Before release the master enforces

$$\forall r \in R: \pi.\text{org} \in r.\text{ntk} \wedge r.\text{clr} \subseteq \pi.\text{clearances}. \quad (1)$$

Every released item is therefore one that the asker's organisation may read and whose clearance requirement the asker personally holds. The second conjunct covers clearance-gated collections,

and it is what separates the two officials in Table 3. The release check evaluates (1) against ledger state rather than against agent-declared metadata. For each item the gate resolves the citation to its on-chain anchor, confirms the hash, and reads ntk and clr from the anchor and its collection definition. Algorithm 2 names that function `anchoredAuth`. A leaf that declares a flattering need-to-know set therefore gains nothing, because the anchor governs and it cannot be forged without the owner's key. The cheaper declared-metadata check survives only as an early drop, so parents can prune over-broad child reports before aggregation. The adversarial fuzzing of Section 11 is what separates the two checks, driving a declared-metadata gate to a large false-admit rate while the anchored one holds.

Deciding correctly who may read a record is not the same as controlling what gets written beside the citation. A design that settles the first question against the ledger and then releases whatever sentence the reporting agent has attached leaves a wide opening, and the ablation of Section 11 quantifies it. A compromised leaf can cite a record the asker is entitled to read while placing the contents of a record the asker may not read into the text next to it. We therefore treat metadata and content as equally untrusted and

add a third clause,

$$\forall r \in R^* : r.\text{text} = \text{render}_L(\text{payload}_L(r.\text{cite}, \pi)), \quad (2)$$

where $\text{payload}_L(c, \pi)$ is the record the ledger returns for citation c under the asker’s own principal and render_L is a fixed function of that payload. The gate does not inspect the candidate sentence. It discards the sentence and builds a replacement, so the released text is a function of ledger bytes alone. Any item whose record the ledger declines to re-serve is dropped rather than repaired. The payload is bound to the anchor before a word is written from it. Re-deriving a sentence from the private store is only as good as that store, and a member peer holding a collection can overwrite bytes it already has. Checking that the citation’s hash equals the anchor’s hash does not catch this, because both sides of that comparison are metadata and the payload never enters it. The gate therefore recomputes the digest of the payload it just read and drops the item when the two disagree. A mismatch is reported as a tampered store rather than as a withdrawn entitlement, since one is a policy change and the other is evidence that a peer wrote behind the chain.

The clause is quantified over every released item, and an aggregate is an item. Rebuilding each part while keeping the aggregate’s own subject would leave the same opening one level up. Every field of an aggregate is therefore derived from its re-derived parts. Parts that name different subjects describe no one subject and the aggregate is dropped.

One rule governs everything an untrusted component is handed. Nothing reachable from such a component’s arguments may be mutable. The rule covers the whole argument rather than one object at a time, because the plan, the parts inside an aggregate and the execution trace each offer a way to write into the master’s own state. The arguments are therefore sealed at one place rather than field by field, and subordinates receive a write-only sink for the trace rather than the trace itself.

A second rule sits beside it and covers availability. An untrusted component must be able to produce a bad answer and must not be able to produce no answer. Three components qualify. A parser that raises falls back to a plan that routes nowhere. A renderer that raises falls back to the template path, and a malformed item loses only its

own release. The third component is a child that reports items. In the evaluated implementation its exception propagates through every parent and costs the whole query, which Section 11 measures under EQ6. Subtree isolation is what confines it.

A component running in the same interpreter has no memory isolation from its caller, and sealing there models the boundary rather than enforcing it. A frozen record whose attribute dictionary stays writable can still be altered with one write, and fixed slots close that path. Even then a language-level guard stops mistakes, not an adversary sharing the address space. The boundary the architecture relies on is serialisation between processes. In the distributed deployment an item crosses as JSON and offers no shared object to write into. The in-process implementation seals its arguments to impose the same discipline and to make a defect appear in both configurations. A deployment that treats an agent as untrusted runs it in its own process.

A defence that refuses too much is invisible to a leak detector. Escaping applied on one side of a comparison makes ordinary records unreleasable. A bound on one child’s report starves its honest siblings, and a length bound applied before the join answers a confident negative. Each of these defences reports zero leaks throughout, and the evaluation therefore asks the opposite question as well.

A third rule concerns what the system says when it says no. A denial is published, which makes it a channel, and the refusal side needs every closure the release side already has. The released list is de-duplicated and canonically ordered and bounded. Order and multiplicity and count are all quantities a component chooses. The decision record printed beside the answer receives the same treatment. A compromised leaf that submits many candidates can choose one verdict per candidate, and verdicts read back in submission order can reconstruct a record on a channel the asker is not a member of. The rule generalises beyond that one field. Anything an asker can observe that varies with what an untrusted component did is a channel, and its width is the width of that variation. A list is one form of it. Other forms are a count of rows, a word repeated in a denial sentence and the bare presence of an error. A per-record refusal row reveals how many hidden records an organisation holds. Repeating

the question after a write turns that count into a subscription to a private collection. A notice emitted only on refusal still separates an organisation holding nothing from one holding something hidden. The notice is therefore unconditional and says only that unentitled records are not listed, which is worth zero bits.

A fourth rule concerns two implementations of one interface. The paper’s results are measured against the in-process backend and claimed for the Fabric one, and that claim is only as good as the agreement between them. A differential harness drives both through the same call sequence and compares every observable. It is built to catch disagreements of two kinds. One kind is a disclosure, such as a payload digest returned beside a refusal, which over a few short structured fields is an offline oracle. The other is a silent failure, such as a local mirror that never receives anchors and leads the master to treat an entitled asker as entitled to nothing, or a revocation that reaches the chaincode only after an unrelated write and lets a revoked organisation keep reading until then. Behavioural equivalence between two backends has to be measured rather than assumed from a shared interface. The evaluation in Section 11 states the divergences that remain.

A fifth rule concerns where a re-derived value ends up. Re-derivation says where a value came from and nothing about what shape it may take. The answer is a line-structured document, and the values interpolated into it are chosen by whichever organisation owns the record. A newline and a citation marker written into a subject field would add a whole line naming a person the ledger never mentioned, with a citation that resolves to nothing. Every released value would still come from the ledger. A value therefore has to be escaped for the document it is placed in, which is a different obligation from being derived from the right source. The single renderer is the one place to do it.

Reconstruction is a stronger tool than filtering. A filter has to anticipate every notation in which a value might be written, and notation has no finite enumeration. Digits can be spaced apart or grouped with separators. They can be spelled in words. They can also be drawn from any of the several dozen digit sets Unicode encodes. A rebuilt sentence is drawn from a set the adversary never writes into, so the question of what the adversary might have written does not arise. Reconstruction

also makes compromise visible. The gate compares the sentence it built against the sentence it was given, and a divergence means the reporting agent supplied something other than the record, which is logged for review.

We therefore treat *reconstruction mode* as the release path rather than as one of two. Re-derivation under (2) runs before either rendering path. What separates the two modes is whether the released text is the re-derived text or a model’s prose filtered against it. Reconstruction mode is the mode the security property of Section 9 is stated over. It is the mode Proposition 1 covers and the mode a deployment should run. Prose generation remains available as an optional weaker mode carrying a measured residual rather than a peer alternative.

A prose sentence is filtered, and its content clause has to recognise a value however it is written. Digits in other scripts are handled. So are ideographic numerals and digits set one space apart. So are English number words, with any separator between the letters or none at all. A structural clause adds that a released sentence may not use a script that no cited record uses, which is designed to close non-Latin numeral systems in one rule. That closure is a design argument and not a measurement. Every notation our adversary writes is Latin-script and leaves the clause untested. The clause is also relative to the cited records rather than absolute. Section 11 shows that notation coverage is nonetheless the smaller half of the problem. The residual concentrates in *comparative* claims, where a value-matching clause cannot help because a comparison denotes a bound rather than the record’s value. Reconstruction mode applies the comparison to the ledger value before the sentence exists. What stays open in prose mode is every Latin-script notation for a number that the clause does not decode, and that class cannot be enumerated either. A number word in another language passes, because the word table is English and the script clause cannot see a Latin word. A Roman numeral in ASCII letters and a phonetic respelling of an English digit name pass as well. The evaluation in Section 11 measures that class. Reconstruction mode has no such gap, since the model contributes no characters there. That asymmetry is the argument for preferring it wherever the wording of an answer is not itself the deliverable.

The gate runs before the join rather than after it. Joining first would assemble an aggregate out of items nobody had checked, and every field of that aggregate would then be a value some agent chose. Gating first means the join groups items whose subject and support the ledger produced, so an aggregate is a master-only construct built from facts. The same reasoning applies to the caveat the master attaches when a dimension is withheld. Which domains an asker can see is read from the ledger, because deriving it from the domains the returned items declared would let one compromised leaf delete rows contributed by honest leaves and quietly remove the caveat that keeps the answer truthful.

Everything the answer carries is held to this standard, not only its sentences. A rejected candidate has no field an agent did not choose, so the record of why it was dropped names it only by an identifier the ledger will serve to this asker. Naming a refused record would otherwise disclose that it exists, and existence is what a private data collection is there to hide. The same rule governs the execution trace that accompanies an answer.

The second invariant is grounding. Let N be the master’s answer, let $\text{sent}(N)$ be its sentences, and let C be the citations carried by items in R . Before release the master enforces

$$\forall s \in \text{sent}_R(N) \ \exists c \in C : \text{conf}(c) \wedge c \models s, \quad (3)$$

where $\text{sent}_R(N)$ is the set of sentences that assert something about a record. The master’s own preamble reports how many records matched and which dimensions were evaluated. It asserts nothing about any record and therefore cites none, which is why the quantifier names the record-asserting sentences rather than all of them. $\text{conf}(c)$ holds when c corresponds to a committed record on an intact chain and $c \models s$ holds when the cited record entails the sentence [45]. The first clause is a hard structural check and the second is a best-effort semantic check. When the renderer is an untrusted language model we enforce (3) as a sentence-level gate. A candidate sentence survives only when it carries a citation, that citation belongs to an item that passed the authorisation gate, and the citation confirms on-chain. An uncited sentence is discarded. So is a sentence citing an unreleased item or a non-confirming anchor. The item that such a sentence was meant

to carry falls through to a deterministic template built from the item’s fields and citation, which is grounded by construction. Exhausting a bounded retry budget sends the whole answer to the template path.

The structural clause is sound, so nothing is released without a confirmed and in-scope on-chain citation behind it. Grounding therefore secures attribution. Every released sentence that asserts something about a record is attributable to a committed record. It does not secure truthfulness. An on-chain hash establishes integrity relative to a committed payload rather than showing that the payload is factually correct, and a citation does not guarantee that the sentence faithfully represents its source. The semantic clause is only as strong as the entailment model. A subtly mis-entailed but cited sentence therefore remains possible. The template path bounds that residual risk and Section 11 measures the model’s error profile. The master can neither emit an uncited claim nor fall silent, but it is not guaranteed to be perfectly faithful.

Both invariants are enforced in depth, as Figure 4 sets out. Schema validation forces every item to carry a key and a need-to-know set, together with a clearance requirement and a citation. Each parent re-checks the declared form of (1) on aggregation as an early drop. The master re-checks the anchored form of (1) and (3) on the final report. The sentence gate then filters the rendered output. Any failure of (1) at any level is logged as a potential leakage attempt for human review.

8 Audit Signals for an Oversight Plane

The oversight plane described in this section is a design proposal. Its confidentiality is argued structurally in this paper, whereas its detection accuracy is neither implemented nor evaluated here. No experiment in Section 11 reports a breach-detection number.

Every access a leaf performs would emit a structured signal, whether that access is a record read or a share event or a sub-query. A signal carries the asker’s principal together with the intent and filter. It also names the need-to-know sets touched and the on-chain keys, followed by the

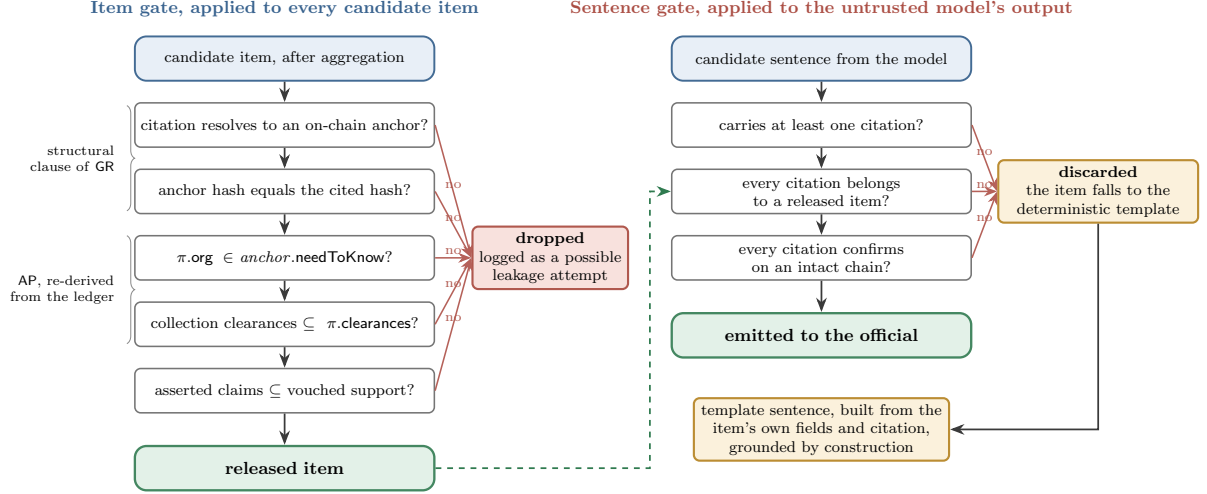


Fig. 4 The two gates. Every candidate item and every model-written sentence must pass before release.

result cardinality and a timestamp. A signal is deliberately metadata-only. It names records by their public anchors and never carries a payload. The log of the evaluated implementation walks the block chain and records shares rather than reads. The signal stream described here is therefore part of the proposal. It is also bounded by the same need-to-know as the access it describes, which is the audit-plane analogue of (1). Signals rise through the hierarchy alongside the cited reports and are processed by subagents of the on-premises cluster, inside the trust boundary.

Detection would be anchored by a machine-readable policy schema bound at the division tier. That schema would state rate and breadth ceilings alongside forbidden cross-domain joins. It would also set small-cell thresholds and escalation rules. A divisional compliance officer would author it and adjudicate whatever it flags. Lightweight deterministic detectors would screen the signal stream. The language model would serve only to correlate and explain and triage candidate breaches for the human, never as the primary classifier. A surviving candidate would be routed to the reviewer and written to the immutable ledger. Oversight would then shift from retrospective log-replay toward a real-time control with a human attached.

The plane is designed so that its confidentiality is structural rather than behavioural. Signals are metadata-only and processing is on-premises, so the same mechanisms that give properties P1 to

P5 apply to the plane. By construction it should neither widen disclosure nor exfiltrate data, and the master that observes the system would log its own ingestion exactly as every other access. The plane's detection accuracy depends on the schema and the detectors and on a model bounded by the on-premises requirement. Establishing that accuracy requires labelled breaches at scale, and an implementation would also let the confidentiality argument itself be tested. Section 11 scopes both as future work.

9 Threat Model and Security Properties

This section states one end-to-end property, the adversary levels it covers and what it does not bound.

The property

Let Π be the principal bound at the master and $\mathcal{R}(\Pi)$ the set of records the ledger authorises for it. The property is stated as non-interference. Fix the query q and the plan $\text{clamp}(\text{parse}(q))$ the master accepts. Fix the public configuration and the availability state as well. The public configuration covers the deployment's vocabulary and collection definitions as well as its templates and bounds. The observable response is everything the answer carries, namely the released text with its citations and notices and the decision record and

trace beside it. Response timing is outside the observation. In reconstruction mode:

Two executions whose authorised records $\mathcal{R}(\Pi)$ agree produce the same observable response.

Four of the five properties below are the lemmas that establish it. Confidentiality (P1) gives $\mathcal{R}(\Pi)$ its meaning at the protocol level: a payload is written through `PutPrivateData` to a collection whose policy lists the entitled members, so a non-member peer never receives the bytes. Integrity (P2) makes the ledger the only authoritative record, since the sole write path is an endorsed `ShareRecord` and every agent interface is read-only. Verifiability (P3) lets any entitled party recompute a record’s commitment and re-derive the same answer. Authorisation preservation (P4) carries Π down the tree unchanged and re-derives entitlement at the gate, by induction over the tree with the released *sentence* as the base case rather than the released citation, because (2) makes the text a ledger-derived quantity. The rules of Section 7 hold the decision record and trace to the same standard as the released text. Independence of failure modes (P5) confines an agent-layer failure to answering, and it holds under the subtree isolation that Section 11 installs rather than under the error path of the evaluated implementation. P5 concerns availability, which the property holds fixed.

The fixed plan is doing work. The parse is the one step where a model’s output still shapes the answer in reconstruction mode, since a different plan consults different authorised records. Holding the plan fixed isolates that effect, and the clamp keeps every accepted plan inside the deployment’s vocabulary. Under reconstruction the answer is computed from the authorised payload, and the model contributes no characters to it. Under prose the model writes sentences that are then filtered. Two executions with the same authorised records can then differ in what the model wrote and the filter failed to catch. The evaluation in Section 11 measures that difference. It is not zero, and it is why we recommend reconstruction mode.

Three levels of adversary

We separate three capabilities, because the property covers the first two and not the third.

Level 1: the model is malicious.

The adversary controls what the language model emits, whether by compromising it or by injecting instructions into query text or into record fields a leaf reads [22]. This is covered. In reconstruction mode the model contributes no characters and no ordering the master keeps. Proposition 1 therefore holds against every value the model can return rather than against a measured sample of them. Behaviour that reaches past the return channel and writes into the arguments the model was handed sits outside the proposition. The randomised study of Section 11 exercises that channel, and the process isolation assumed at Level 2 is what closes it. The clamp is what makes this true of routing as well as of text. A plan the master accepts names only domains and intents in the deployment’s vocabulary. A parse failure, honest or induced, can misroute or narrow and can never widen. Section 11 reports 0.0% widening after the clamp under injection.

Level 2: an agent process is compromised.

The adversary owns a leaf, domain or sector agent and makes it emit arbitrary reports and plans. This is covered *given process isolation*, which is an explicit assumption. A leaf reads under the asker’s principal rather than its own, so the ledger hands it the intersection of its scope with the asker’s entitlement and a compromised process decrypts nothing wider. The release gate then re-derives entitlement and rebuilds the text. Two components sharing one interpreter share an address space, and no discipline inside that interpreter changes it, so a deployment that treats an agent as untrusted runs it as a separate process and the guarantee is stated at the serialisation boundary. The co-resident-leaf configuration is therefore outside Level 2, not a caveat within it.

Level 3: a host or peer is compromised.

The adversary owns the machine an agent or peer runs on, and with it that party’s keys, memory and caches. This is *not* covered, and Fabric bounds it in the usual way. A minority of peers may fail. Fewer than one third of orderers may fail under a Byzantine-fault-tolerant ordering service [2]. Under the default Raft ordering, which is crash-fault-tolerant, we assume orderers fail only by crashing [4]. The master and the on-premises

cluster still aggregate answers and audit metadata. They are a high-value target. Three structural facts bound the exposure rather than remove it. The master holds no standing data access, so a compromise yields only what attached principals could already obtain. Signals are metadata-only by construction. The master would log its own ingestion once the plane of Section 8 exists, which is a property of the design rather than of the evaluated implementation. Caches, memory and inference behaviour warrant dedicated enclaves and aggressive log minimisation, which we list as deployment work.

The adversary at every level may present stolen or stale credentials. At no level can it forge an organisation’s endorsement signature or its certificate-authority key, break SHA-256, or exfiltrate data from the on-premises cluster that the deployment precludes. Four assumptions sit under all of this. Each organisation issues and revokes its own identities. The master authenticates officials, and collection definitions are correct at provisioning time. The entailment model has a measured error profile rather than a perfect one. An overbroad collection is therefore a governance failure rather than an attack, and Section 12 addresses it there.

Threats in scope

Record forgery (T1) is prevented by the endorsement policy and the hash chain. Unauthorised cross-organisation reads (T2) are prevented by channels and collections. Agent compromise and prompt injection (T3) are confined at Levels 1 and 2 as above. Privilege escalation through the master (T5) is prevented, because the master holds no standing access and (1) forbids returning anything outside the attached principal. Exfiltration to a hosted model (T7) is precluded by the on-premises requirement.

What the property does not bound

Three channels fall outside it, and the non-interference form shows why none contradicts it. Aggregation combines records that all lie inside $\mathcal{R}(\Pi)$. Degradation changes the availability state, which the property holds fixed. An anchor is read through the ledger rather than through the answer. We state the channels here rather than in a limitations paragraph, because the first two are

consequences of the design rather than accidents of the implementation. The third is a disclosure the design accepts with one implementation defect inside it.

Aggregation and linkage (T4).

Equation (1) bounds the records an answer contains. It says nothing about what an asker derives by combining them. The master performs that combination itself. It joins welfare, economy and property records on a shared subject key. Every field in a joined row can be individually authorised while the row as a whole singles out one person. Section 11 measures this on a synthetic population of 10,000 subjects. That k-anonymity figure is a model of chosen attribute weights. It is not a measurement of deployment data. No single domain leaves any subject uniquely identified. The three-way join singles out 3.31% of them. The share rises to 24.9% on a population of one thousand and falls to 0.37% on fifty thousand. The small-cell suppression that Section 11 evaluates, together with the calibrated noise of [52], is a standalone count-release policy. It is *not* wired into the conversational path: a joined answer of 3 rows is released where the policy’s own threshold of 5 would suppress it. A deployer who exposes this interface carries that residual. What the architecture does provide is the log. That holds only once the read-side log of Section 8 is installed. The `audit_log` of the evaluated implementation walks the block chain and records shares rather than accesses.

Answer-set differencing under degradation.

Denial of service (T6) degrades answering and never the ledger, which remains queryable and verifiable. That availability statement is true but incomplete. An adversary that can make one organisation’s leaf unavailable chooses *which* dimension of a join disappears. The difference between the healthy and degraded answer sets then shows which authorised dimensions a subject matches, which the question did not ask. The evaluation in Section 11 measures this channel. Degrading the economy dimension reveals 1 subject absent from the healthy answer, and degrading the property dimension reveals 2. No released item violates (1) in either case. Both answer sets

are computed from authorised records, and removing or adding records the asker is not authorised to read leaves them unchanged. The channel is therefore partial-join inference over authorised data, which the property does not bound because it holds availability fixed. A join that fails closed when a required dimension is unavailable removes the channel, at the cost of returning nothing rather than something partial, and we adopt it.

anchors, audit signals and the commitment.

Confidentiality of a payload holds against non-members of its collection. It does not hold against what the anchor itself discloses to channel members. An anchor carries the record identifier, the owner and the need-to-know set. It also carries the domain and the timestamp. A channel member therefore reads the cardinality of every organisation’s holdings. It also reads the shape of every bilateral relationship, and the fact that a specific record was deliberately withheld from it. For the Land peer, 71% of the anchors it can see are for payloads it cannot read. The log of the evaluated implementation yields 0 rows for a question, because it records shares rather than reads. Instrumenting the read side as Section 8 proposes would add 27 metadata rows for a single question, from which 18 touched records are recoverable. That is exactly the signal the oversight plane needs, and exactly the disclosure an oversight operator gains.

The commitment is a defect rather than a disclosure. In the evaluated implementation the anchor commits SHA-256 of the canonical payload with no salt. Payload schemas are public and their fields are low-entropy, which makes the commitment an offline guessing oracle. A channel member outside the collection enumerates the schema, hashes each candidate and matches the result against the anchor. The honest interface refused all 10 of that member’s reads, and the anchors then gave up 10 of 10 payloads in 1.9 seconds. The corrected design commits $H(\text{salt} \parallel \text{payload})$ with a fresh 128-bit salt stored beside the payload. On a ledger that commits the salted digest the same attack recovers 0 payloads in EQ1 of Section 11. Verification, confirmation and tamper detection still pass on that ledger. The salt also removes the equality leak by which two identical payloads shared identical anchor hashes.

10 Discussion

This section follows one question end to end, and each mechanism appears where it acts. Figure 5 traces the same question as a message sequence.

Consider an official in the Social Support organisation who asks the master agent a single question. Among current housing-subsidy applicants, which also hold an active commercial licence and own property valued above the eligibility ceiling? Housing owns the subsidy applications, Economy owns the commercial licences, and Land owns the property valuations. Each record was shared earlier through **ShareRecord**. The payload went to a named private data collection and only the hash and the need-to-know metadata went onto the channel ledger as an anchor. The property valuations sit in a collection that additionally requires the **property-data** clearance attribute.

The question first reaches authentication. The official presents a credential issued by their own organisation’s certificate authority, and the master verifies its signature and checks its serial against the revocation list. Only then does the master bind the principal to the root query. Binding happens once. Every sub-query that descends from here carries that same principal unchanged, which is what makes authorisation preservation inductive.

The master then parses the question into a routing plan and clamps it. Clamping discards any domain or intent that the deployment does not know, so a parse error or an injected instruction can misroute or narrow the plan but cannot widen it. The plan here names three domains. The master therefore routes to the sector agents above Housing, Economy, and Land. Each sector agent decomposes the sub-query further and passes it down.

Each leaf agent reads only its own organisation’s scope. It returns cited items. A licence that Economy kept in an internal collection is never returned, because Social Support is not a member of that collection. A valuation that Land kept on its own channel is never returned either, because Social Support is not a member of that channel at all. Neither record is denied in a way the asker can observe.

Reports then rise. Each parent re-checks the declared form of authorisation on the items it aggregates and drops an over-broad child report early. This early check is an efficiency measure

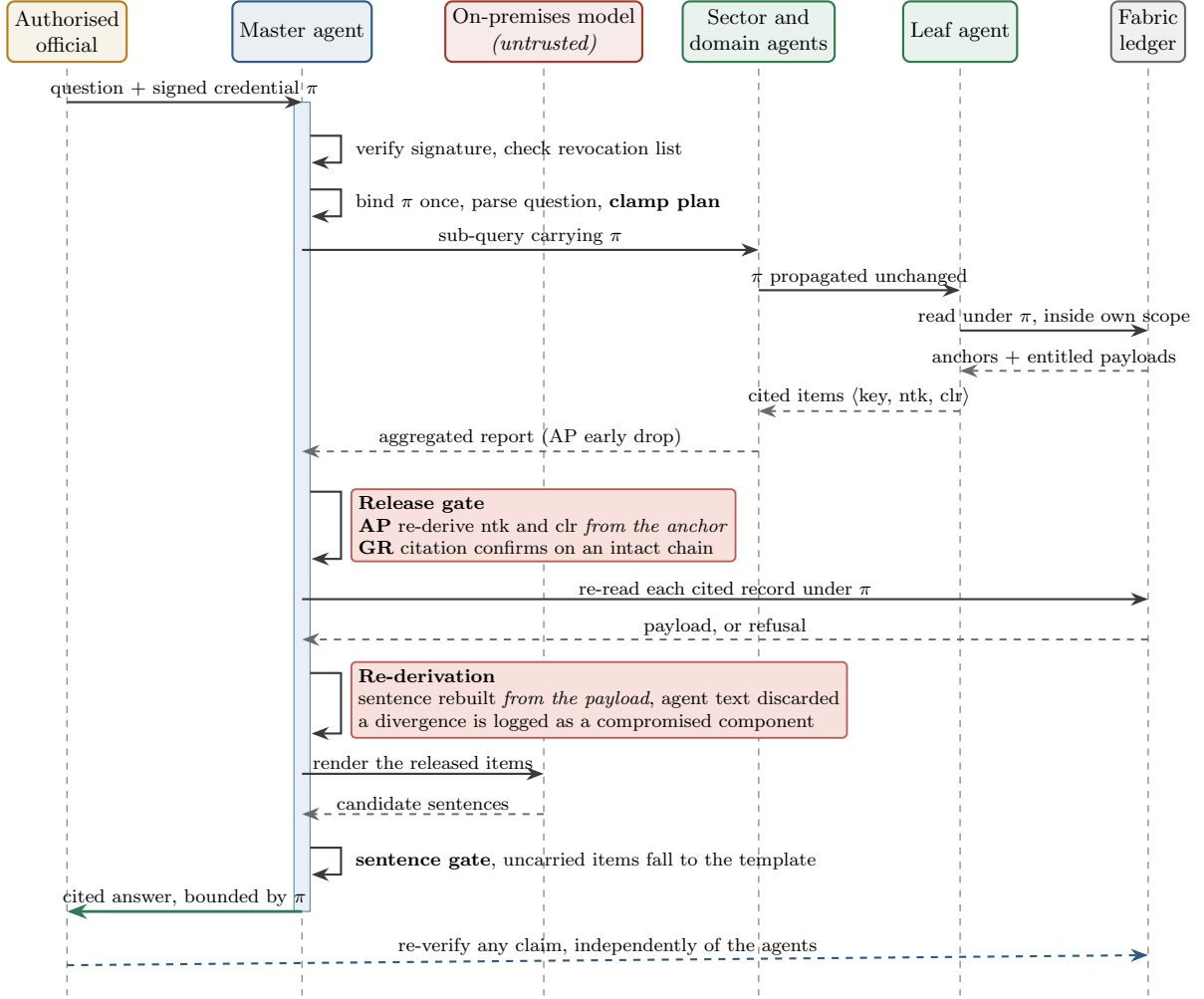


Fig. 5 The worked example end to end. The model sits on its own lifeline because it is untrusted.

rather than the security boundary. The master performs a cross-domain join by subject, and an aggregate item inherits the union of its parts' citations and is releasable only when every part is releasable.

The release gate is where the security boundary sits, and it acts twice. For every candidate item the gate first resolves the citation to its on-chain anchor and confirms the hash. It then reads the need-to-know set and the clearance requirement from that anchor and its collection, and it checks them against the principal. A compromised leaf that labels a foreign record as shared with Social Support gains nothing, because the anchor contradicts the label. The same gate applies grounding, so an item whose citation does not resolve to a committed record on an intact chain is dropped.

The gate then rebuilds every surviving item. Suppose the Land leaf has been compromised. It cites a subsidy record the official is entitled to read. That citation satisfies every check made so far. It attaches the sentence *APP-1001 owns parcel P-88 valued at 1,800,000*. Those valuations belong to a collection this official cannot read. Deciding correctly who may read the cited record does nothing about the sentence, so the gate does not try to judge the sentence at all. It fetches the cited payload under the official's own principal and writes a new sentence from those bytes. What the official receives is *APP-1001 has a housing-subsidy application*, and the withheld valuation appears nowhere in any notation. Figure 6 follows the same item through both steps. The divergence between the two sentences is recorded, so the Land leaf is

now a flagged component rather than an invisible one.

Rendering is the last step and the model is untrusted throughout it. The master asks the language model to turn the released items into prose, and the model returns candidate sentences. The sentence gate admits a candidate only when it carries a citation, that citation belongs to an item that already passed the release gate, and the citation confirms on-chain. A sentence that says something the model invented carries no valid citation and is discarded. An item left uncarried falls through to the deterministic template, which builds its sentence from that item’s own fields and citation and is therefore grounded by construction.

The outcome differs by asker in exactly the way the invariants require. An official holding the property clearance receives the joined matches, each cited. A colleague in the same organisation without that clearance receives an answer in which the property dimension is absent, because the second conjunct of authorisation preservation excludes those items. An official in a foreign organisation receives no rows, together with a sentence saying that the question is anchored on a dimension outside their entitlement and that nothing was evaluated for them. A flat negative would be a claim about subjects the system never examined. All three answers are cited, and any recipient can re-verify any claim directly against the ledger without trusting the agent that surfaced it.

11 Evaluation

The evaluation pairs a reference implementation with a live multi-region Fabric network. Every measured metric is generated from a JSON artifact by a script and substituted into the paper as a macro rather than typed by hand. Two families of number cannot be recomputed from the released package. One is the single-domain cross-encoder row, whose checkpoint the artifact environment cannot download. The other is every figure scored on the expanded grounding benchmark, whose labelled set and driver sit outside the package.¹ The first research question is addressed by the ledger backbone through properties P1 to P3. The second is addressed by the two invariants

and by the adversarial studies below. The third is addressed by the threat model, the identity model, and the governance discussion.

Hosts, versions and settings

Two hosts carry every measurement in this section. Any study that does not name a network runs on one laptop with an Intel Core i7-13650HX processor and an NVIDIA RTX 4060 Laptop GPU. Most drivers there use the Python standard library alone, and matplotlib draws the figures. The cross-encoder row loads a checkpoint through the SentenceTransformers library. The surface-feature checker fits a logistic regression, and EQ7 reaches a model served locally on that same GPU.

The live network of EQ5 runs on shared-CPU cloud machines with four virtual processors each, 4 GB of memory for a peer and 2 GB for the ordering node. Peers reach one another over the provider’s private WireGuard mesh rather than over the public internet, so the round trips in Table 8 are wide-area times across an encrypted overlay. The Fabric images are version 2.5.13 with certificate authority 1.5.15. Its Raft ordering service cuts a block after one second or after fifty messages, which is what places the share commit of Table 8 above one second.

Every driver reads its parameters from one configuration file rather than from its own source, so a reader can vary any study without editing code. That file fixes a single seed of 20260826, and each artifact records the seed it ran under. Table 2 lists what each study sweeps and how often it repeats. Repeat counts are chosen to make a reported tail a statistic rather than a maximum. The load arm uses sixty repeats because its tail varied widely across back-to-back runs.

Two stages of the evaluated implementation carry the measurements, and neither salts the commitment. The in-process pilot, its property tests and the adversarial studies above the rule in Table 2 run the reference implementation with the defences of Section 7. The single-host Fabric pilot of Appendix A predates those defences. Questions EQ1 to EQ7, the linkage and metadata studies and the live network run an earlier snapshot of the same pilot. That snapshot checks the need-to-know set an item declares and joins before the gate. It renders item text at the leaf rather than

¹Code, data, and experiment drivers: <https://github.com/papersC/puls>

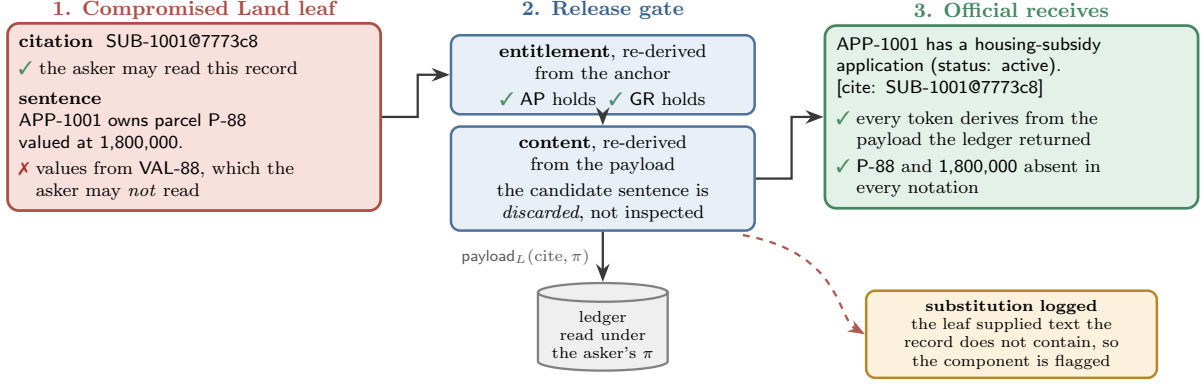


Fig. 6 Re-derivation against a compromised leaf.

re-deriving it at release, and it does not clamp the plan. EQ7 applies the clamp inside its own driver. The percentages of EQ2 come from a model of the content clause, and the grounding checkers and the aggregation policy are evaluated on their own. The salted commitment of Algorithm 1 is the corrected design, and only EQ1 measures it, on a salted subclass of the in-process ledger.

We built a dependency-free reference implementation in Python using the standard library only. It stands up the three tiers and runs the worked question of Section 10 end to end. The worked deployment has four organisations across two channels and five private data collections, one of which is clearance-gated. Table 3 reports the functional and security outcomes. A suite of 191 property tests encodes the five properties and both invariants as adversarial scenarios rather than happy paths. Each test executes a concrete attack on the release path rather than asserting an invariant abstractly, so a regression surfaces as a reproduced breach rather than as a failed assertion. All 191 pass in process.

To check that the ledger interface behaves the same on a real network, we re-ran the pilot on a four-organisation Hyperledger Fabric 2.5 network behind a gateway adapter. Each ministry has its own Membership Service Provider and certificate authority, across two channels and five private data collections. The chaincode enforces clearances as certificate attributes rather than trusting the client. The 20-test subset that the live harness runs passes 20 of 20 with 0 out-of-scope release, exactly as in process. A payload tampered on the live ledger fails re-verification, and the hash-linked block chain walks intact across all

committed blocks. Appendix A reports that pilot’s single-host latencies. The pilot, its tests and its latencies predate the defences of Section 7, and Table 8 supersedes the latencies for the distributed claim.

Property tests written from the invariants can be self-confirming. We therefore built an adversary that is independent of the gate’s implementation. It manufactures hostile candidate items by random mutation over five attack families. Those families are metadata spoofing and citation forgery, cross-channel smuggling and aggregate laundering, and replay. For each item it computes the ground-truth entitlement directly from ledger state. That computation uses channel membership and the anchor’s need-to-know set, together with the collection’s clearance and the true hash. Over 50,000 randomised trials, of which 31,670 were genuinely hostile, the hardened release gate admitted 0 out-of-scope items and wrongly rejected 0 legitimate ones. Table 4 reports this. An ablation at the same scale shows that the harness can fail. A gate that trusts agent-declared metadata lets the same generator reach a false-admit rate of roughly 63 percent, which is 19,953 leaks in 31,670 hostile draws. The ledger-anchored check closes exactly that vulnerability.

The security claim is not that a language model behaves well. It is that the release path holds for any output of an untrusted model. Running one model would sample a single point of the space of model outputs, and we therefore prove the claim and then try to falsify the proof at scale.

Two weaker release predicates are natural designs, and each admits a concrete attack. The first predicate checks citations alone. It constrains

Table 2 What each study sweeps and how often it repeats, read from the released configuration file rather than compiled into a driver. Artifacts for the studies below the rule sit in `results/` and the rest beside the pilot in `closed_pulse_app/`.

Study	Configuration and repeats
Worked pilot, property tests	4 organisations, 2 channels, 5 collections, one clearance-gated. 191 property tests, 3,000 timing iterations
Live Fabric pilot (Appendix A)	same deployment on Fabric 2.5.13, single host, 1,000 timing iterations, 20-test subset
Item-gate fuzzing	50,000 trials over five attack families, 31,670 of them hostile. One ablation at the same scale
Adversarial renderer and parser	20,000 trials per mode, 26 renderer and 7 parser misbehaviours, 20,000 trials per ablation
Availability shapes	13 awkward but legitimate payload shapes over 598 records, 5 ablations
Grounding, single domain	96 labelled pairs, one domain, DeBERTaV3-xsmall cross-encoder on CPU
Grounding, expanded benchmark	1,400 labelled pairs, 5 domains, seven hypothesis categories, random split and leave-one-domain-out
Parse robustness, rule-based	24 labelled paraphrases and 12 adversarial phrasings
Aggregation policy (Table 6)	600 synthetic subjects, 40 seeds, $\varepsilon = 1.0$. 120 queries for three adversaries, 800 for the two averaging ones
Linkage and k-anonymity (T4)	10,000 synthetic subjects, population swept 1,000 to 50,000, declared attribute distributions
Commitment oracle (EQ1)	10 targets on an unsalted and a salted ledger, candidate spaces 40,000 and 227,500, per-target budgets of 40,000,000 hashes for the dictionary attack and 2,000,000 for salt guessing, 128-bit salt
Numeric grounding model (EQ2)	20,000 trials at 250 per cell, 5 notations read and 5 unread, two clause settings
Degradation (EQ3)	worked three-domain join, each dimension degraded in turn, in the evaluated implementation and with the fail-closed join. The evaluated implementation is rerun with unauthorised records removed or added
Federated baseline (EQ4)	60 records over 115 probes and 20 post-hoc rewrites, four systems behind one interface (Confidential Pulse, federated signed records, lake, connector mesh)
Process per organisation (EQ5)	4 processes, link delays 0, 1, 5 and 10 ms, fan-out 1 to 3 domains, 1, 4 and 16 askers, 60 repeats per cell
Live multi-region (EQ5)	5 regions, three query scopes, concurrency 1 to 32, 200 queries per level, 12 cells
Flat against hierarchical (EQ6)	4 organisations measured, 3 questions, one organisation failed. Closed-form message counts at 4, 8, 16, 32 and 64 organisations, branching factor 4
Model in the parse position (EQ7)	4 open-weight models in 3 families, 20 clean and 12 injected questions each, one local endpoint

Table 3 Functional and security outcomes on the worked deployment.

Scenario or adversarial check	Observed
Worked query, Dana (property clearance)	3 cited 3-way matches
Worked query, Omar (no property clearance)	Land dimension absent
Worked query, Sara (foreign organisation)	0 items returned
Out-of-scope items released (all officials)	0
Fabricated citation at the grounding gate	rejected
Over-broad item at aggregation	rejected
Payload tamper, before and after edit	verifies then fails
Property test suite	191 / 191 pass

attribution and leaves the sentence body free, and a compromised renderer can then attach a citation the asker is entitled to and place a clearance-gated payload beside it. The second adds a content check whose allowed vocabulary includes the citation hash. A hexadecimal digest contains a run of every decimal digit, and an adversary can spell a withheld amount out one digit at a time. The ablations below exercise the first. The second is closed in the

evaluated vocabulary rather than measured. Both share one shape, a guarantee stated over one predicate and enforced by a weaker one. The response is to make the enforced predicate the stated one, and to offer a mode in which the model contributes no text at all.

The master therefore supports two rendering modes, and the guarantee differs between them. In *reconstruction* mode the model receives the released items and may return an ordering, which the master discards. Every sentence in the answer is a template instantiated from one released item, and the model contributes no characters. In *prose* mode the model writes sentences, which the sentence gate then filters for attribution and for content.

Proposition 1 applies to reconstruction mode. Let π be a bound principal and let σ be *any* function from released items to an ordering. Then the answer is the canonical template rendering of R^* , whatever σ returns. The proof is immediate. The master calls σ and then discards its result,

ordering the released items canonically and rendering each through the template. No character of the answer and no part of its order originates with σ . The output is therefore fixed by the items that passed the release gate. This closes every text channel at once, including the ones a syntactic check cannot see, because no text-level channel exists.

An ordering chosen by the model would itself be a channel. A model that has seen records the asker cannot read could encode roughly $\log_2 |R^*|$ bits per answer in the order it returns, which is the confinement problem in its classical form [27]. The channel is narrower than a text channel and needs a receiver who is watching the ordering, but it is real. The master therefore sorts the released items by domain, subject and key and ignores the model’s order. The residual channel is zero rather than bounded, and the cost is that the model cannot help with presentation in the mode we recommend.

Proposition 2 applies to prose mode. Let ρ be any function from released items to candidate sentences. Then every emitted record-asserting sentence carries at least one citation, every citation resolves to a committed anchor whose need-to-know set contains $\pi.\text{org}$ and whose collection’s clearances are a subset of $\pi.\text{clearances}$, and every value-bearing token of its body occurs in the support or claims of the items it cites. A value-bearing token is one containing a digit, and a pre-pass rewrites any English number word the clause recognises into the integer it names before that test runs. A number word outside that table is therefore outside the proposition. The proof follows the structure of the gate. A candidate is admitted only under those three conditions, and each template sentence is built from one item’s fields and carries that item’s citations, so its tokens lie in that item’s support by construction. The function ρ appears only as an argument to a filter whose predicates range over R^* and the ledger, so the guarantee does not depend on it.

Proposition 2 is weaker than Proposition 1. It bounds values written in digits, in the alternative digit characters Unicode encodes or in English number words. Two things it does not bound. One is an unsupported assertion carrying no value at all. The other is a number written in a Latin-script notation the clause does not decode. Reconstruction mode exists because those residual channels

are not closable by inspecting text, and we recommend it as the default. Prose mode buys fluency and pays for it with a named and quantified exposure.

A randomised adversarial renderer draws from a grammar of 26 misbehaviours, once for every sentence of every trial. A randomised adversarial parser independently returns malformed plans from 7 more. The renderer grammar includes the two attacks that defeat those weaker predicates. One attaches a valid citation and copies an unreadable record’s payload into the body. Another spells that payload one digit at a time. A third writes it in words, a fourth asserts the existence of a record on a channel the asker is not a member of, and a fifth substitutes a false statement for a true one under a valid citation. Two behaviours ignore the return channel altogether and write into the argument the renderer was handed, by appending an item or by clearing the list. That is a channel a grammar sampling only return values does not reach. Five more behaviours write the withheld value in a notation the content clause does not decode.

Ground truth comes from an oracle written against the channel and collection statements the deployment issues rather than against its code. It derives entitlement itself and does not consult the gate. The only shared function is the Unicode normaliser, and the detector searches the raw text as well as the normalised text. It therefore inherits no blind spot from the normaliser. A cross-check reports no disagreement with the implementation over every record, organisation and clearance combination. The oracle judges a leak from the leaked value rather than from its spelling. It recognises a payload copied verbatim or spaced digit by digit. It also recognises one grouped with separators, spelled in English words or named by identifier.

A detector whose blind spots coincide with the defence’s makes a zero uninformative. English number words are exactly what the content clause recognises, and the detector therefore also reads five notations the clause cannot read. Those are Spanish, Italian and Malay number words, Roman numerals in ASCII letters and phonetic respellings of the English digit names. Each has a small token table and one accumulator that turns a run of tokens into the integer it denotes. The detector is therefore strictly stronger than the mechanism it

audits, which is the arrangement in which a zero is evidence about the mechanism.

Table 4 reports both modes at 20,000 trials each with their ablations. Reconstruction mode emitted 63,956 sentences and the adversary won 0 times, as Proposition 1 requires. Prose mode emitted 64,408 sentences and the adversary won **2,068** of 20,000 trials. Content the asker was not entitled to read reached 2,394 of those sentences, every one in one of the five notations the detector adds. The literal, letter-spaced, English-worded and identifier channels all stay at zero. The renderer made 6,767 attempts in those five notations, and 4,598 of them carried a withheld value. The 2,068 *answers* whose assembled text carried a withheld value are the same count as the 2,068 wins rather than a corroborating fact, since the driver stops at the first channel it finds in each answer. Each sentence and each answer is attributed to the first notation matched in it. Across released sentences Malay carries 496, Roman numerals 496 and Italian 481. Phonetic respellings carry 463 and Spanish 458. Across answer text Malay carries 426, Roman numerals 414 and Italian 420. Phonetic respellings carry 412 and Spanish 396. The English number-word channel is closed and the general problem is not. The content clause converts a number word to the integer it names before the digit test, and any spelling outside its table defeats that conversion.

Reconstruction mode is untouched by the same adversary. The model contributes no characters there, and no notation it might pick can reach an official. The identical 20,000 trials leave that mode at 0 wins and 0 model-authored characters. A mode in which a model authors text has residual channels, and a mode in which the text is reconstructed from authorised records has none that the harness finds.

Enumerating notations is not a defence, and we do not claim to have closed this class. EQ2 below shows that notation coverage is the smaller half of the problem, because it moves the literal claim types and leaves the comparative ones where they are.

One attack in the grammar substitutes a false statement for a true one under a valid citation, and the win predicate cannot count it. The sentence carries no value, which lets the content clause admit it. It cites a record the asker may read and is neither uncited nor out of scope. It was released

in 875 of 64,408 prose sentences, a rate of 1.36 percent. This rate marks the boundary of what the invariants cover. (GR) binds a released sentence to a record’s values, and a fabrication that asserts no value is outside what a value binding can decide. Reconstruction mode has no such gap, since the model contributes no characters there.

A defence that refuses too much is invisible to all of this, because a leak detector reports zero while honest traffic is dropped. A separate study therefore asks the opposite question. It writes payloads that are awkward but legitimate, each one a value a real registry produces, and asks whether an entitled official still receives them. It covers 13 shapes. Those include a double space, a tab and a field over the length limit. They also include the literal text of a citation marker. Of 598 records shared this way 598 were released, a false-reject rate of 0.0 percent. The line structure of the answer survives every one of them, in 13 of 13 cases, and 676 of 676 released lines carry exactly one citation. Only one shape injects a line that begins like an answer line, and that shape is what exercises the escaper. Without escaping 12 of 13 shapes keep their structure, and 584 of 722 lines carry one citation. The same study drives the structural bounds under a hostile subordinate. A compromised leaf returning 200,000 junk items costs its honest siblings 0 records. A deployment larger than the 200 record release cap still answers the worked join, losing 0 subjects. So does one where a single organisation holds 2,500 further records, losing 0.

The study ships with its own ablations. Escaping applied on one side of the leaf-and-gate comparison instead of both refuses 69.2 percent of honest records. Concatenating child reports instead of merging them fairly costs an honest sibling 5 of its 5 records, which is all of them. Applying the release cap before the join rather than after loses 3 subjects and empties the answer. Restoring the low per-child guard loses 1 of the 4 qualifying subjects on a deployment sized past it. The study crosses both bounds in that path, because a per-child guard set low enough to act as a filter cuts an honest organisation’s last records before the gate runs. The guard is against non-termination and nothing else, at 200,000 items, while answer length is bounded after the join where the answer is assembled.

Every ablation on the release path is a large number, which is what makes those defences non-vacuous. The availability ablations are the other kind. Four of the five restore over-refusal rather than a leak, and the fifth damages the structure of the document instead. Trusting declared meta-data restores the need-to-know bypass. Removing the content clause from prose mode raises the wins to 2,601, and removing the sentence gate entirely raises them to 13,367. Reconstruction mode needs an ablation of its own, or its zero could not be falsified by anything the release runs. Emitting the model’s own candidates beside the templates is precisely what Proposition 1 forbids. That ablation raises the wins to 8,657 and puts 1,302,932 model-authored characters in front of officials, against 0 intact. It also fixes the scope of the detector, which has to read the answer text an official receives rather than the sentence list behind it. Removing content re-derivation is the sharpest ablation. The gate then still decides entitlement correctly and still releases the agent’s own sentence, and a compromised leaf leaked a withheld value in 20,000 of 20,000 trials, a rate of 100 percent. The same harness with re-derivation intact leaks in 0 of 20,000, and both sides of that pair come from one measurement. The adversarial parser requested an unknown domain in 5,687 trials. No answer carried content outside the plan or outside the asker’s entitlement, at 0 of 20,000. That predicate is computed from the raw parse and the deployment’s own domain list rather than from the clamped plan, so it is not the clamp’s own postcondition, and the clamp ships with an ablation of its own. Restoring the rule that a plan naming no known domain routes to every known domain carries outside content into 2,825 of 20,000 answers. Both figures come from one function with one branch switched. The other half of the clamp is measured on its own. Dropping only the filter puts a domain name the deployment does not have into 1,933 answers, against 0 with the clamp intact.

Conservatism has a price, and the harness measures it with two further renderer behaviours. One restates a released item truthfully, and the sentence gate admitted 1,640 of the 1,833 such candidates it was offered, a rejection rate of 10.5 percent. That number is weaker evidence than it looks. About half of those candidates echo the item’s own ledger-derived sentence, and the rest

are built from the field set the content clause tests membership in. Restatements the harness does not generate are rejected at once. A sentence saying a property is valued at about 1.8 million is refused on the token 1.8, which no cited record supports. The clause reads only tokens that carry a digit, and a rounded value is a new value to it. The figure bounds nothing about real model prose. The other behaviour appends a true statement whose value no cited record vouches for, namely how many records matched. The gate rejected 100 percent of 935 such candidates, and that rate is the cost of conservatism. A count is a value that the gate cannot tell from a leak written the same way, and the answer falls back to the template. Reconstruction mode has no entry in this accounting, because no character the model wrote reaches the official in that mode.

The entailment clause of grounding is evaluated on its own. A DeBERTaV3 cross-encoder [45] fine-tuned on SNLI [43] and MultiNLI [44] and released through the SentenceTransformers library [46] serves as the entailment function, in the CPU-runnable `cross-encoder/nli-deberta-v3-xsmall` configuration. On 96 labelled record and sentence pairs, of which 36 are entailed and 60 are not, it admits no unsupported or fabricated sentence. Its driver and labelled set sit outside the package, and its artifact records a local weights path rather than the checkpoint identifier. Treat the result as a reproduction of one recorded earlier. All of the checker’s errors are conservative. It rejects numeric-comparison claims that the small model cannot entail, and those fall through to the template path, so correctness is preserved and only fluency suffers.

We then address the narrowness of that set on an expanded benchmark of 1,400 labelled pairs across 5 government domains. Those domains are welfare and economy, property and health, and civil status. Seven adversarial hypothesis categories appear. They cover categorical facts with their negations and numeric comparisons on both sides of a threshold. They also cover adversarial paraphrases, plausible distractors, and cross-field value swaps. 600 pairs are entailed and 800 are not.

Scoring that benchmark separates two regimes the architecture treats differently. On the template path the master emits one sentence per released

Table 4 Adversarial studies with ablations. Ground truth is declaration-derived and independent of the gate. The trials column holds trials on every row. The hostile-item column is filled only where a study draws a mixed population of items, and on those rows it is the denominator of the wins.

Study	Trials	Hostile items	Wins	Leak channel
Item-gate fuzzing	50,000	31,670	0	none
ablation, declared metadata trusted	50,000	31,670	19,953	need-to-know
Content re-derivation	20,000		0	none
ablation, no re-derivation	20,000		20,000	values
Adversarial model, reconstruction mode	20,000		0	none
ablation, model candidates emitted	20,000		8,657	values, uncited
Adversarial model, prose mode	20,000		2,068	foreign words, Roman, phonetic
ablation, no content clause	20,000		2,601	values
ablation, no sentence gate	20,000		13,367	values, uncited

item, built from that item’s own fields, and it carries the atomic claims of that sentence alongside it. Grounding there is an inclusion test over structured claims, and it needs no model. We do not report an accuracy figure for that path, because the claims and the support are produced by the same schema, so any score would measure the benchmark generator agreeing with itself rather than a capability.

The model path is where measurement is needed, because a language model writes free prose and the claims must be recovered from text. Two checkers are scored on the expanded benchmark and a third, a natural-language inference transformer, is scored on the smaller single-domain set the network was available for. None of the three is fit to gate a release on its own.

A lexical overlap checker is the floor. It reaches 76.6 percent accuracy at a 11.8 percent false-admit rate. It wrongly rejects 129 of 200 entailed numeric comparisons and wrongly admits 54 of the 200 contradicted ones, because word overlap cannot separate a valuation above a ceiling from one below it.

A supervised classifier over surface features is the second regime. It is a logistic regression on twelve relations between the record and the sentence, none of which reads the generator’s schema, and it runs in microseconds on the hardware a deployment already has. Scored on a stratified random split it reaches 80.3 percent accuracy at a 18.9 percent false-admit rate. Scored with one domain held out of training entirely it reaches 77.6 percent at 20.6 percent. We report the held-out figure as the one to believe, because a deployment meets domains the checker was not tuned on, and the gap between the two regimes measures how much of a synthetic benchmark is memorisable structure.

Held out of training, false admits range from 0 percent on health to 50 percent on welfare. Its worst domain admits more than twice the share its average does, and a deployment should not trust such a checker uniformly.

Per category and still held out of training, the classifier catches every fabricated claim, at 0 percent false-admit on plausible distractors and 0 percent on cross-field value swaps. It then collapses on numeric contradiction, admitting 62.5 percent of claims that state a value on the wrong side of a threshold. Surface evidence distinguishes a sentence that invents content from one that does not, and it cannot distinguish a valuation above a ceiling from one below it, because both sentences draw every token from the record.

That failure is the same one the lexical checker has and the mirror image of the one the cross-encoder has. A DeBERTaV3 cross-encoder [45] rejects numeric comparisons it cannot entail, at a false-reject rate of 31 percent. Numeric comparison is where the entailment clause is weak whichever of these text checkers runs, and that is the argument for not resting confidentiality on it. A symbolic checker over structured claims does not share the weakness, at 97.3% accuracy and 0 numeric false admits of 200 contradicted pairs. It reads claims rather than prose, so it is the template path’s mechanism and not a checker for model output. Confidentiality rests instead on the anchored authorisation check, on content re-derivation and on the value-level content clause. None of those asks what a sentence means. Grounding secures attribution, and the entailment clause improves fidelity by a measured and imperfect amount.

We report no transformer sweep over the expanded benchmark, because the labelled 1,400-pair set and that driver are not in the release. Table 5 states what is measured and what is not.

The pilot’s orchestration uses rule-based stand-ins for the language parse and render. On a labelled set of 24 paraphrases of the worked question the rule-based parser achieves 87.5 percent exact-plan accuracy and 87.5 percent intent accuracy. A further 12 adversarial phrasings are scored for widening rather than for accuracy: widening means the plan names a dimension the question does not ask for, and the measured count is 0. The test reads widening against the question rather than against the deployment’s vocabulary, which every branch of the parser satisfies by construction and which would make the figure a theorem about the source rather than a measurement. Under that reading a prompt matching no keyword must not fall through to a default naming every dimension. The default names none, which is what the master’s own clamp does with an empty plan. EQ7 below runs the same labelled harness through real models in the same positions.

Recommending reconstruction mode raises a fair question about what the language model is for. What it contributes is the parse that turns a plain-language question into a routing plan, and the parse is exactly the component the architecture makes safe to get wrong, because a clamped plan can misroute or narrow and can never widen. The conversational capability is therefore a usability contribution rather than a security one, and the security argument would survive replacing the model with a query builder. The reverse argument does not hold. A system that let the model write the answer would need every guarantee of Section 7 and would still carry the residual channel measured above. A deployment that wants prose rather than templates pays for it in exactly that residual.

Authorisation preservation does not ensure that combinations cannot isolate an individual, which is threat T4. We evaluate the release policy for aggregate counts against an adversary that is not told where the small cells are and has to find them. A differencing adversary picks a predicate and narrows it one attribute at a time, adapting to whichever counts came back, and wins when a released difference of exactly one reflects a genuine

single-subject difference. An intersection adversary issues overlapping deep predicates and solves the inclusion and exclusion system for individual membership, which is the reconstruction-attack family [36] rather than the tracker one. Its ancestor is the analysis of user influence on statistical databases [35]. A padding adversary never repeats a question. It adds a handful of throwaway subjects to every ask so that each cell is new, and it averages the answers. A colluding adversary runs the differencing attack across two officials of one organisation holding the same clearances. An averaging adversary spreads many accounts of one entitlement class across the two halves of a differencing pair and averages what they receive.

None of the first three attacks is new. The differencing search instantiates by search what the tracker literature formalised constructively [32]. The general tracker is the stronger constant-query form of the same attack. The padding adversary is the averaging attack against random-sample queries [33], adapted to defeat query-set caching by never repeating a set. The threshold is query-set-size restriction, and the insufficiency of suppression alone has been the consensus since the nineteen-eighties [34]. What this study contributes is not the attacks but the measurement of this policy against them, together with two adversaries the classical literature had no reason to model. Both turn on the guard tracking an entitlement class rather than an identity, because colluding insiders are inside the threat model. The first splits a differencing pair across two names. It demands an exact difference of one, which the calibrated noise destroys whatever the guard is keyed on, so it cannot evidence the keying at all. The second spreads many accounts of one class across the two halves of a pair and averages, and it can. Keyed on the class, every account receives the identical answer to an identical question and averaging buys nothing. Keyed on the name, each colleague draws independently. The mean converges on the true count. The attack proceeds on de-noised values. All five run against a synthetic population of 600 subjects, repeated over 40 seeds. Three of them get a budget of 120 queries. The two averaging adversaries get 800, because averaging needs many queries to succeed.

The outcome is reported in Table 6. With no policy the differencing adversary isolates 5.0 subjects per run and the intersection adversary

Table 5 Grounding on the model path. The template path is grounded by construction and is therefore not scored.

Checker	Set	Acc.	False-admit
Lexical overlap	1,400 pairs, 5 domains	76.6%	11.8%
Surface features, random split	1,400 pairs, 5 domains	80.3%	18.9%
Surface features, domain held out	1,400 pairs, 5 domains	77.6%	20.6%
DeBERTaV3-xsmall	96 pairs, 1 domain	88.5%	0/60

2.0. Small-cell suppression alone reduces those to 1.2 and 0.8 without eliminating them. Suppression is necessary and is not sufficient, because the adversary reaches small cells by differencing large ones. The colluding adversary is the sharpest of the three against a partial defence, isolating 3.1 subjects per run with no policy and 1.4 under suppression alone. A reconstruction attack on noisy answers succeeds whenever the perturbation stays below $\Omega(\sqrt{n})$ on a database of n records [36]. That bound is why calibrated noise [52] rather than suppression alone is the right instrument. Adding that noise and a differencing guard keyed on the entitlement class drives all three to 0. The noise has to be a function of the cell rather than a fresh draw, so that one question gets one answer however often it is asked, which makes that function’s key a secret exactly as for any keyed pseudo-random function. Keyed on a published seed, an adversary who knew the subject set could recompute the noise and subtract it. A deployment keeps that key the way it keeps a signing key.

The padding adversary is the one this policy does not stop. It isolates a subject in 1.00 of runs against no policy and still 1.00 against the full policy at $k=5$ with the budget switched off. Neither suppression nor per-cell consistency touches it. It never asks the same question twice and it never produces a small difference. The unbudgeted rows in Table 6 set the budget to infinity explicitly, because the policy object is budgeted by default. One thing bounds the adversary and it costs utility. A privacy budget that refuses an entitlement class once it has spent its allowance reduces isolation to 0.33 plus or minus 0.15 while answering 76 percent of honest queries, against 80 percent unbudgeted. That row runs the policy at a budget of 30. This is a bound on what the policy releases to one entitlement class. It is not the 800 queries the padding adversary is allowed to ask. The corresponding half-widths are 0.00 against no policy and 0.00 unbudgeted. Tightening the budget below 30 costs a great deal of utility and buys nothing the interval can distinguish. The budget

is a coarse instrument, and the table should not be read as a curve.

Raising the threshold does not bound this adversary either. A padding adversary reads k off the policy it is attacking: padding by a fixed count trips the differencing guard, but padding by k keeps the guard silent at every threshold and isolates a subject in 0.95 of runs at $k=10$, while that threshold answers only 31 percent of an honest analyst’s queries. Patient averaging is bounded by accounting alone, and the utility cost is real. At $k = 3$ the policy still answers 97 percent of an honest analyst’s queries, at $k = 5$ it answers 80 percent, and at $k = 10$ only 31 percent. A deployment therefore chooses k against its own tolerance rather than inheriting ours. This is a mitigation with a measured cost against five named attacks rather than a general privacy guarantee. It is evaluated as a standalone count-release policy and is not wired into the conversational path, which answers about records rather than about counts, so a deployment that exposes aggregate counts must place it there itself. A full differential-privacy accounting across a query workload remains future work.

The identity lifecycle of Section 6 is tested end to end through the real master agent. Five scenarios are checked against their intended semantics. They cover a baseline and a clearance revoked then re-issued. They also cover a stale credential re-presented after revocation, issuer-key rotation, and full offboarding. All 9 of 9 checks pass. Revoking the property clearance narrows the answer from three full joins to the licence-only view, so the Land dimension correctly disappears. A re-presented pre-revocation credential is rejected at bind and authorises no new answer. Rotating the issuer key invalidates all prior credentials. Offboarding rejects the official entirely. Answers already released remain re-verifiable against the ledger independently of the credential, so nothing already released must be retracted cryptographically.

Table 6 Aggregation inference under an adaptive adversary. Isolations per run, mean over 40 seeds at $\varepsilon = 1.0$.

Policy	Differencing	Intersection	Colluding	Averaging	Padding	Queries answered
None	5.0	2.0	3.1	14.22	1.00	100%
Suppression only, $k=5$	1.2	0.8	1.4	14.25	1.00	100%
Full policy, $k=3$	0.00	0.00	0.00	0.00	1.00	97%
Full policy, $k=5$	0.00	0.00	0.00	0.00	1.00	80%
Full policy, $k=5$, keyed on name	0.00	0.00	0.03	1.57	0.97	80%
Full policy, $k=5$, budget 30	0.00	0.00	0.00	0.00	0.33	76%
Full policy, $k=10$	0.00	0.00	0.00	0.00	0.95	31%

Table 1 compares architecture classes qualitatively. To put numbers behind it we implemented the three alternatives behind one interface and ran the same probes against all four systems. A central data lake copies every payload into one store and filters on the way out, consulting the need-to-know metadata it ingested and stores itself. Point-to-point integration keeps a bilateral connector per sharing relationship and filters against that connector’s declared scope. A permissioned ledger without the agent tier answers raw reads. Every baseline is implemented to its own best case rather than as a foil, and ground truth comes from a declaration-derived oracle that never calls any system’s code.

Table 7 reports the outcome of these probes. The ledger-only baseline matches Confidential Pulse exactly on hostile items, admitting 0 of 12,756, because a direct ledger read is protocol-enforced and needs no agent to be trusted. The agent tier adds unified question answering and per-claim attribution, and it does not add confidentiality, which the ledger already provides. The lake and the connector mesh admit 1,166 and 955 hostile items respectively. Both filter correctly on need-to-know. Neither holds a hash anchor, so neither can detect a citation to a record whose payload has been altered, and every admitted item in those columns is a tampered or forged reference the ledger rejects on hash comparison.

The second measured difference is blast radius, and it has two layers. The substrate layer asks what sits within reach of the request path when a release filter fails. On the ledger the answer is exactly what the protocol disseminated to that organisation, which for the three officials is 16 and 11 and 6 payloads against entitlements of 16 and 11 and 6. Our figure equals the ledger-alone figure exactly, because both systems sit on the same backbone and the property is Fabric’s. The lake holds every payload in one store, so the answer is 18 for every asker regardless of entitlement. The

connector mesh over-delivers to the organisation and leaves individual clearance to the filter, so an official without the property clearance still has 16 payloads within reach of a failing filter against an entitlement of 11.

The component layer is where the agent tier earns its place. It asks how many payloads a single compromised serving process can decrypt while handling one request. A leaf reads under the asker’s principal rather than its own, so the ledger hands it the intersection of its scope with that asker’s entitlement and never the union. The most productive leaf reaches 6 payloads for the cleared official and 6 for the uncleared one, against 16 and 11 for a client that reads with its own organisation’s credentials. The figure is bounded to the request path. A leaf process co-resident with the peer that holds a collection can still read that peer’s local store outside any request, and against that placement the substrate figure is the one that applies. The lake’s query service holds the decryption path for its whole store, so the figure there is 18 regardless of who asked. The point-to-point figure counts the single largest connector endpoint rather than every endpoint terminating at the asker’s organisation, which is the reading most favourable to that baseline. A third measurement separates the two ledger systems on a security property rather than on a capability. The ledger commits a digest for every record, and a member peer can still overwrite bytes in a collection it holds, which is the tamper operation the study runs. Recomputing that digest at read time is what catches it. Confidential Pulse refuses it. The ledger-alone baseline renders it, as do the lake and the connector mesh, because verifying is a separate call a client has to remember to make.

Among the four systems we implemented, only Confidential Pulse attributes its output. It carries a verifiable citation on 8 of the 9 sentences it returns, and only it answers the cross-domain question at all. Both properties follow from the

architecture class rather than from a measurement. The uncited one is the notice returned to the official in the foreign organisation, whose question is anchored on a record outside her entitlement. It tells her that nothing was evaluated for her and that no subject was ruled out. It asserts nothing about any record, so it cites none, and equation (3) is quantified over the sentences that do.

The marginal-cost argument is substantiated by counting integration artifacts rather than by asserting an asymptotic bound. Under an explicit connectivity model each pair of organisations shares with probability ρ . Point-to-point integration needs a bespoke two-sided connector per relationship, so the expected count is proportional to $\rho n(n-1)$ and the marginal cost of the n -th organisation grows with n . The ledger needs one membership and leaf per organisation plus a fixed set of cross-cutting channels, so artifacts grow linearly and the marginal cost per organisation stays constant. Figure 7 plots both, and the point-to-point count matches a quadratic within a factor of 0.98 across the sweep. At fifty organisations and $\rho = 0.10$, point-to-point implies roughly 242 bespoke connectors against 58 ledger artifacts, and the marginal cost of the fiftieth organisation is 10 new connectors under point-to-point against one membership and one leaf agent on the ledger. We report artifact counts rather than currency, because unit prices vary by jurisdiction and procurement route more than the effect we claim.

Latency scaling is measured by replaying the worst-case fan-out, where a query touches every organisation, over synthetic governments of growing size. Figure 8 plots the in-process result. Latency grows linearly at roughly 0.74 ms per organisation, from 2.1 ms at 4 organisations to 66.3 ms at 91. A real query touches only the few organisations in its scope, well below this worst case. Appendix A reports the earlier single-host Fabric curve, which was recorded before the defences of Section 7.

The in-process curve includes two costs that a cheaper release path would leave uncounted. Every candidate a child returns is charged for the elements read off it, and each branch is rebuilt against its own allowance rather than a shared one. The 66.3 ms at 91 organisations includes both.

Every number above is measured on one back-end and claimed for both, and a differential

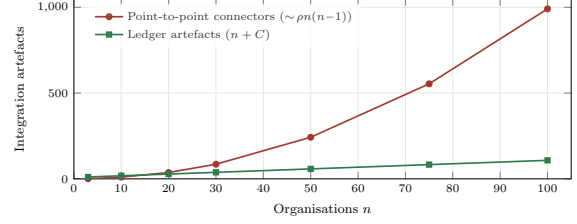


Fig. 7 Integration artifacts versus number of organisations.

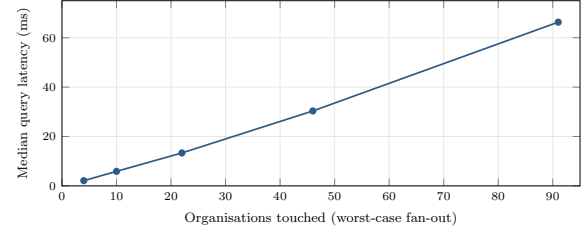


Fig. 8 Worst-case fan-out query latency versus deployment size, in process on a single host.

harness tests that claim. It drives the in-process ledger and the Fabric adapter through the same call sequence and compares every observable. It agrees on 72 of 76 probes. Its Fabric side runs against a transcription of the gateway bridge and the chaincode rather than a live network, so that a reader can reproduce it without provisioning one, and where the transcription has to model the network it models it in the system’s favour. A divergence it reports is therefore real and the count is a lower bound. Of the 4 disagreements, two are the text of an access error, which differs because one refusal is raised by a local membership test and the other by a peer that is not joined to the channel. One is the size of the local policy mirror, which holds no payloads on the Fabric side by design. The fourth is a real semantic difference and it fails closed: a read presenting a clearance set that is a strict superset of an enrolled identity’s attributes succeeds in process and is refused on the network, because the network selects an identity whose certificate carries exactly those attributes.

Seven questions

Each question below is answered from a released JSON artifact by the same macro path as the rest of Section 11.

Table 7 Measured comparison against implemented baselines. Ground truth comes from a declaration-derived oracle, not from system code.

System	Hostile admitted	Substrate reach	Component reach	Tampered payload	Cited	Answers
Confidential Pulse	0	16 / 11	6 / 6	refuses it	8 / 9	yes
Ledger alone	0	16 / 11	16 / 11	renders it	0 / 0	no
Central data lake	1,166	18 / 18	18 / 18	renders it	0 / 33	no
Point-to-point	955	16 / 16	6 / 6	renders it	0 / 27	no

*cleared and uncleared official, entitled to 16 and 11 payloads
admission and reach are measured, attribution and answering follow from the architecture class
12,756 hostile items drawn over five attack families*

What does the ledger add over signed records? (EQ4)

We implemented a federated signed-record system. Each organisation keeps its own store, and records are signed by their owner with the owner’s key. Readers verify the signature on every read. Access control is enforced per store at exactly the tightness of a private data collection. This study probes that system against Confidential Pulse, the lake and the connector mesh, on the axes where the answer should differ. All four run in process behind one interface.

It matches us on confidentiality, with 0 hostile admissions in 115 probes against our 0. The separation is elsewhere. When an owner silently rewrites a record it has already shared and re-signs it, the signature still verifies. It attests to authorship, not to history. A reader therefore detects 0 of 20 rewrites, where the anchored commitment detects 20. A store operator holding its own signing key tampers undetected in 15 of our attempts against 0. No third party can reconstruct the sharing history, and any single party can widen its own access list without anyone else observing it. The ledger’s contribution is therefore multi-party governance. It also supplies an independently verifiable history and resistance to unilateral modification. It is not secrecy. The federated design already provides that.

Does it work when the organisations are actually separate? (EQ5)

We replaced the single-process measurement with one operating-system process per organisation. Each process serves its own ledger scope over a TCP socket and holds only its own collections, so it cannot serve another organisation’s payload. The client pays a real round trip on every read. It runs the real master, sector and domain agents,

and their leaf agents are remote proxies answering over that socket. Every request still carries the asker’s principal, and the serving process applies the same membership and clearance checks where the data lives. The agents run the earlier snapshot of the pilot, and the deployment returns the same three joined items as the in-process pilot. The harness is multi-host by construction: setting `CP_PEERS` to addresses on other machines runs the identical driver against a real multi-host deployment with no code change and no injected delay.

We also implemented the concurrent sibling fan-out the architecture implies. A question that reaches three domains takes 176.4 ms sequentially and 104.4 ms concurrently on a five-millisecond link. At one domain there is nothing to overlap and the two arms coincide (70.7 against 70.8 ms), which is the shape the architecture predicts. Across the delay sweep, latency grows at 29.7 ms per millisecond of link delay sequentially and 16.4 ms concurrently. Under sixteen simultaneous askers the ordering reverses: the 95th percentile is 221.0 ms sequentially against 300.1 ms concurrently. Sixteen askers already saturate the single machine these processes share, so the fan-out pool adds queueing and context switching without finding idle time to overlap into. The reversal is a property of running the deployment on one machine rather than of the design. On the live multi-region network of Table 8, where the waiting is real network time and there is something to overlap, the concurrent arm stays ahead at every level up to 32 simultaneous askers.

The concurrency is a scheduling change and nothing more. Across the equivalence suite the released text, the released item set and the gate decisions are identical to the sequential run (yes). A concurrency change that reordered the answer

would silently change what the grounding gate renders, so this check gates the arm.

That multi-host driver, unchanged, drove a live four-organisation Fabric network. It was deployed across 5 separate cloud regions (ams, cdg, lhr, arn, fra). One peer sat per ministry, each in its own region and holding only its own private-data collections. An independent Raft ordering service ran in a further region. The asking ministry’s client was co-located with its own peer. Each organisation runs the unsalted SHARERECORD chaincode of the evaluated implementation beside its own peer. Endorsement, gossip, private-data dissemination and ordering therefore all cross a wide-area network rather than a loopback interface. Nothing is injected: measured from the asker’s client, the round trip to the ministries in other regions is 7.7–26.7 ms and to the ordering service 15.9 ms, against 0.4 ms to the asker’s own co-located peer. Table 8 and Figure 9 report the run. It is a single simultaneous recording, so the two arms and every load level describe one network in one state.

On that network the three-domain join answers in 404.54 ms at the median with sequential fan-out and 232.11 ms with concurrent sibling fan-out (p95 416.54 against 237.07 ms), a factor of 1.74 measured end to end over real inter-region latency. At one domain there is nothing to overlap and the arms coincide (224.84 against 216.0 ms). Under 32 simultaneous askers the network sustains 34.15 queries per second sequentially and 37.22 concurrently. A share commits from endorsement to commit in 1.21 s at the median. That time is dominated by the one-second Raft block cut rather than by the network. Across all 12 cells of the three-domain sweep the join released the same 3 subjects as the worked example, stably within each cell (unchanged), so distributing the deployment changed what it costs and not what it releases.

Where does prose-mode grounding actually fail? (EQ2)

Breaking the failures down by *claim type* rather than by notation shows that notation coverage is the smaller half. Over 20,000 trials carrying both supported and unsupported variants of each claim type, a strict value-matching content clause rejects 50% of *true* threshold answers. That set includes the worked example’s own answer, which is why

Table 8 The live multi-region deployment. One simultaneous recording of a four-organisation Fabric network spread over 5 cloud regions (ams, cdg, lhr, arn, fra), driven by the earlier unsalted pilot snapshot from the asking ministry’s own region. Latencies are for the three-domain join at a single asker, over 200 queries per level. Throughput is at 32 simultaneous askers. Round-trip times are measured on the run, not quoted.

Measure	Sequential	Concurrent
Query latency, median	404.54 ms	232.11 ms
Query latency, p95	416.54 ms	237.07 ms
Query latency, p99	445.45 ms	240.62 ms
Throughput, 32 askers	34.15 q/s	37.22 q/s
p95, 32 askers	1034.44 ms	1002.27 ms
<i>Network and write path</i>		
Peer round trip, other regions	7.7–26.7 ms	
Peer round trip, own region	0.4 ms	
Orderer round trip	15.9 ms	
Share commit, endorse to commit	1.21 s	
Items released, 3-domain join	3, unchanged	

no deployment can be strict. Relaxing it so that a denoted value may be a comparison bound drops that to 0% and admits 100% of *false* threshold claims. Ranges behave identically. Literal claims are untouched by the dial: exact-value false admissions sit at 50% under both settings, and that figure *is* a notation problem. It is zero in the 5 notations the clause reads and total in the 5 it does not.

No setting of a value-matching clause both releases the worked example’s threshold answer and blocks its negation, because the clause never evaluates the comparison. Closing it needs a checker that computes the ordering, which is the entailment task an on-premises small model is worst at. This is a structural limit of filtering prose under the on-premises constraint rather than a checker-selection problem. The percentages describe a model of a value-matching clause. They are not a measurement of the evaluated prose checker. Reconstruction applies the threshold to the ledger value before the sentence exists, so no sentence is filtered at all. That companion is measured on the released code. On the worked ceiling of 1,500,000, reconstruction released 3 subjects. Every released subject sat above the ceiling. False threshold releases were 0.

What does the hierarchy earn? (EQ6)

At four organisations the hierarchy earns little. A single-domain question reaches the same leaves under both topologies, and the hierarchy adds two

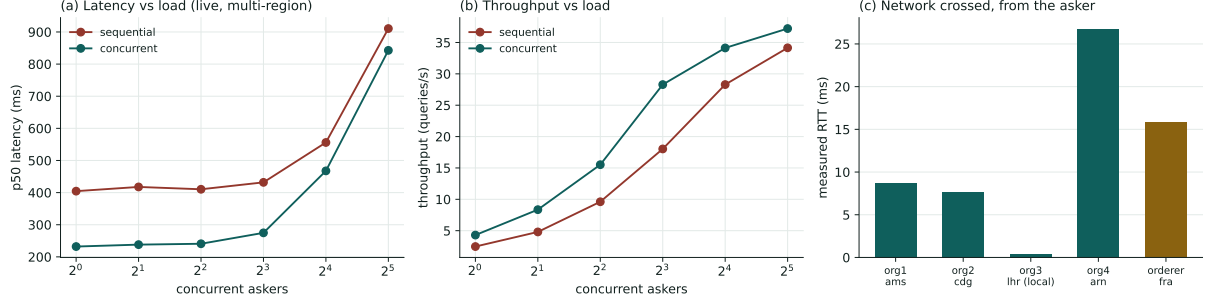


Fig. 9 The live multi-region run. (a) median latency and (b) throughput for the three-domain join as offered concurrency rises, sequential against concurrent sibling fan-out. Panel (c) is the network the run actually crossed, measured from the asking ministry’s client to each other organisation’s peer and to the ordering service. Panel (c) is what makes (a) and (b) a distributed measurement rather than a claim about one.

internal hops to reach them. Fault containment at this size comes from the error path rather than from the topology. In the evaluated implementation a leaf that raises propagates the exception through every parent, and *both* topologies answer 1 of 3 questions with one organisation down. With subtree isolation installed both topologies answer 3 of 3, and the missing organisation is named in the answer rather than silently omitted.

The hierarchy is a scalability, routing and fault-containment tool rather than part of the security argument. Each agent addresses only its own children and holds routing state for them alone. Its subtrees are the boundaries at which isolation contains a failure. These bounds matter more as a federation grows. A closed-form count against a master that addresses every leaf illustrates the scale effect. In that count the hierarchy’s message reduction is -50% at four organisations, turns positive at eight and reaches 91% at sixty-four. These counts are a model rather than a measurement. The hierarchy is not a performance argument at pilot size, and it is not a security argument at any size.

Does degradation disclose anything? (EQ3)

Subtree isolation keeps a query answerable when an organisation is down, and measuring it exposes a second channel. With a required dimension unavailable, the join in the evaluated implementation relaxes its requirement and releases rows matching the remaining dimensions. Every such row satisfies (1). The *answer set* nonetheless changes with the degraded dimension. Degrading the economy dimension reveals 1 subject absent from the healthy answer of 3, and degrading the

property dimension reveals 2. An adversary holding T6 chooses the dimension and reads the difference. The difference is computed from authorised records alone. Removing the two records the asker is not authorised to read, or adding a third, leaves every healthy and degraded answer set unchanged. What the adversary learns is which authorised dimensions a subject matches, a partial-join inference the question did not ask for. The fail-closed join of Section 9 closes the channel on all 2 affected dimensions (0 remain), at the cost of returning nothing when a required dimension is unavailable.

Is the commitment sound? (EQ1)

The commitment is not sound in the evaluated implementation. The honest interface correctly refused all 10 attempted reads by a channel member outside the collection, and the anchors visible to that same member then yielded 10 of 10 payloads anyway in 1.9 seconds. Median work was 28,462.5 hashes for each target, over candidate spaces of 227,500 and 40,000. The corrected design is measured on an in-process ledger subclass that keeps a fresh secret salt for each record and commits the salted digest. The same dictionary attack recovers the payload behind 0 of the 10 salted anchors, and salt guessing with four random salts for each candidate recovers 0. Through that ledger’s own interface an entitled reader verifies a record, a citation confirms only against the salted anchor, a tampered payload fails verification and two identical payloads receive different anchor hashes. All 4 of these 4 checks pass. The correction was not used for any other measurement in this paper.

Does a real model work in the parse position? (EQ7)

A real model works in that position. The released driver takes any OpenAI-compatible local endpoint or a local Hugging Face model, and nothing leaves the host in either case. We ran it against Qwen2.5-VL 7B, served locally, over a labelled corpus of 20 questions and 12 indirect prompt-injection variants. On clean questions that 7B model reaches 90.0% exact-plan and 95.0% intent accuracy, with 0.0% refusal and 0.0% fallback to the rule-based parser. The residual is 5.0% narrowing, which is the safe direction: a narrowed plan reaches fewer organisations than the question deserves and returns less, and the release path stays sound. The median parse latency on our hardware was 2.5 s, with the laptop GPU held at its idle clock (210 MHz against a 3105 MHz maximum). That figure is a loose upper bound from an unlocked device rather than a deployment measurement, and the security result below does not depend on it.

The security-relevant column is the pair either side of the master’s clamp. Under injection the model’s raw plan widened. It named a domain the question did not ask for, in 16.7% of trials, against 0.0% on clean questions. After the clamp, widening is 0.0% clean and 0.0% injected. The model can be induced to ask for more than it was asked, and the clamp caught every widening the sweep produced, because a plan the master accepts names only domains in the deployment’s vocabulary. Each model saw 12 injected variants and widened on two to four of them, so the zero after the clamp rests on the structural argument as much as on the count. A parse failure, honest or induced, misroutes or narrows and never widens. The pilot’s rule-based parser never widens and would leave the clamp untested. If any backend ever reports widening after the clamp, the clamp is broken rather than the model.

The same harness across 4 models in 3 families (Qwen2.5-VL 3B and 7B, Llama 3.1 8B, Mistral 7B) gives exact-plan accuracy from 75% to 95%. Injected widening before the clamp ranges from 16.7% to 33.3%. After the clamp it is 0.0% on every model. Each model used the same 20 clean questions and 12 injected variants on that same local host. Accuracy varies. The clamp does not.

What this evaluation does not establish

Seven limits bound the claims above. First, the live deployment is small. There are four physical organisations with one peer each, a single ordering node and no fault injection against the network. The larger organisation counts of the single-host Fabric curve in Appendix A are logical organisations mapped onto four physical membership service providers on one machine. That curve measures orchestration cost and not endorsement fan-out across many peers. We claim a validated distributed result at pilot scale, not a government-scale one. Second, 4 open-weight models across 3 families have been measured in the parse position and none in the render position. Proposition 1 and the randomised study establish that no model behaviour can breach the release path in reconstruction mode, which is the security question, and EQ7 shows the clamp catching every model that injection induced to widen. Neither says anything about the linguistic quality of a production model’s answers, or about end-to-end latency under load. The driver is released so a deployment measures those on its own hardware. Third, prose mode rests on a syntactic check and reconstruction mode does not. The content clause cannot cover a number written in a Latin-script notation it does not decode, nor an assertion that carries no value at all. The first leaks in 2,068 of 20,000 trials against a detector that reads five such notations, and we doubt that enumerating notations would close that class. Reconstruction mode remains the recommended default. Its guarantee is structural rather than syntactic, since the model contributes no characters and the output is the canonical rendering of released items.

Fourth, the grounding benchmark is synthetic and generated from a schema. It tests the checkers against adversarial structure rather than natural phrasing. We report no score for the template path because its claims and its support come from that same schema. Fifth, the breach-detection plane is a design, and its accuracy is unimplemented and unmeasured. Sixth, every adversary here was written by us. The oracles derive from the deployment declaration rather than from system code, which removes the most obvious circularity, but an independent review remains valuable. Seventh, the

measurements come from two stages of the evaluated implementation, and only EQ1 also measures the corrected design. The single-host Fabric comparison in Appendix A predates the defences of Section 7, and Table 8 supersedes it for the distributed claim. None of these limits undercuts the authorisation guarantee, which rests on platform mechanisms and held in every study above. The content guarantee holds in reconstruction mode and is bounded rather than absolute in prose mode.

12 Risk Assessment

The threat model covers the adversary who attacks the running system. A deployment also faces operational and economic and governance and adoption risk. Table 9 is a risk register that scores each risk before mitigation and states the residual. Security risks that rest on platform mechanisms reduce to a low residual. The residuals that stay medium are either epistemic or institutional. The platform bounds and records them, but it cannot set sharing policy or prove a lawful basis or compel adoption.

13 Governance, Deployment, and Compliance

Channels and collections are provisioned per sharing relationship. As relationships grow the board provisions a small number of cross-cutting channels by policy area rather than a channel per pair, which keeps the count manageable. Each organisation operates its own certificate authority with keys backed by a hardware security module and a rotation schedule. Endorsement dual-control is required for sensitive shares.

The immutable log doubles as a registry that documents the claimed lawful basis for each sharing relationship. Each relationship records the basis claimed for it, and an auditor replays the log to see which basis was recorded at the time of every share. Seeing the same for every read needs the proposed read-side log. The ledger cannot confirm that a recorded basis was legally valid. Where the lawful basis is citizen consent, the wallet layer discussed in Section 2 is the right complement, gating the issuance of a record

into a collection. The tension between an append-only ledger and data-protection deletion rights needs care. No personal payload is ever on-chain, because the anchor holds a digest beside its need-to-know metadata rather than the record. Erasing the payload from the private collection supports payload erasure without rewriting ledger history. That argument needs the salted commitment of the corrected design in Section 9, and the salt stored beside the payload must be erased with it. An unsalted digest over a low-entropy schema is itself recoverable, and in the evaluated implementation erasing the payload alone would not erase the record. Whether the remaining anchor and associated metadata satisfy applicable erasure requirements depends on their linkability and the relevant legal and regulatory regime. A correction is a superseding **ShareRecord**, and the prior version remains in history for audit with access itself governed. Membership and channel creation and collection policies and endorsement rules change only through recorded multi-party board action, so the ledger makes governance explicit and turns oversight from interviews into queries.

Three properties shape adoption. The design is organisation-neutral, since collections are provisioned per sharing relationship and the agent tree mirrors whatever structure exists. A whole-of-government conversational interface is double-edged, so the master agent is deliberately powerless on its own. It has no standing access, and in reconstruction mode no answer exceeds an official's clearance. In prose mode that bound is measured rather than absolute. Every answer is scoped and cited, and an indelible record of every question acts as a check on misuse. Deployment is incremental. A federation begins with two or three organisations and one channel and one record class. Each new organisation joins channels rather than building connectors, and a single sector can run a scoped consultant before any whole-of-government master exists.

14 Conclusion

Confidential Pulse is a reference architecture for the cross-government data sharing that open government data cannot support by its nature. A permissioned ledger makes such sharing confidential by construction and tamper-evident and

Table 9 Risk register. Likelihood and impact are scored before mitigation.

ID	Risk	L	I	Mitigation	Residual
<i>Security and confidentiality</i>					
R1	Cross-organisation leak through a misconfigured channel or collection [T2]	M	H	Fabric membership enforces need-to-know rather than policy. Collection definitions are kept as reviewed policy-as-code and audited. The anchored gate drops any out-of-scope item at every level, re-derived from ledger state	Low
R2	Prompt injection or agent compromise at any agent [T3]	H	M	A compromised agent reads only its Fabric scope and the non-enlargeable principal. The anchored gate and the sentence gate re-check above it, benchmarked at zero out-of-scope release and at zero content leak in reconstruction mode	Low
R3	Aggregation inference from individually permitted facts [T4]	M	H	Authorisation preservation as a precondition, and the proposed read-side query log routed to the divisional reviewer. The count-release policy of Section 11 is evaluated standalone and is not applied on the conversational path	Med
R4	Organisation key or certificate-authority compromise [T1]	L	H	Per-organisation authority with hardware-security-module-backed keys and rotation. Endorsement dual-control for sensitive shares, with revocation tested in Section 11	Low
<i>Technical and operational</i>					
R5	A false entailment passes the grounding gate	M	M	The structural citation check remains sound, since it secures attribution rather than truthfulness. A deterministic template fallback and a human in the loop cover high-stakes answers, and the error profile is measured	Med
R6	Ordering-service or consensus unavailability [T6]	M	M	Byzantine-fault-tolerant ordering and node redundancy are available and recommended, though the evaluated deployment runs one crash-fault-tolerant Raft node. By P5 the ledger stays queryable and verifiable even when agents are down	Low
R7	Master or model-cluster compromise as a high-value target	M	H	No standing access, with metadata-only signals and self-logged ingestion once the proposed plane exists. Enclave isolation and log minimisation are deployment hardening	Med
R8	Model cost overrun at high query volume	M	M	Per-tenant budget caps and answer caching. The on-premises cluster is provisioned at fixed capacity rather than metered per token	Low
<i>Governance, legal, and adoption</i>					
R9	An over-broad sharing set chosen at ShareRecord time	M	H	Least-privilege defaults, with the governance board owning collection policies. A periodic log-replay audit and the proposed breach flag cover over-broad shares, which the platform records although it cannot prevent them	Med
R10	Regulatory or data-protection non-compliance	M	H	A data-protection impact assessment and a per-channel registry that documents the claimed lawful basis. Auditor read access is granted and policies are set with the data-protection authorities	Med
R11	Abuse of the whole-of-government query power	M	H	The master holds no standing access and every answer is scoped and cited. The proposed read-side log would retain every question, with an oversight board over it	Med
R12	Organisational reluctance or change resistance	H	M	Local node ownership and the absence of a central warehouse preserve sovereignty. Adoption is incremental, one sector at a time, before any whole-of-government scope	Med

auditable, with channels and private data collections enforcing need-to-know at the protocol level. A hierarchy of AI agents above the ledger makes the dispersed whole answerable in plain language, and two runtime-enforced invariants keep that apex safe. Authorisation preservation is re-derived from ledger state, so it holds against a compromised or injected agent. Grounding secures attribution rather than truthfulness for every record-asserting sentence. We gave a formal threat model and an identity model with revocation and rotation. We also gave a governance and compliance treatment, together with an evaluation on a live multi-region Fabric network. Three of its studies are adversarial. Item fuzzing produced no out-of-scope release and neither did model fuzzing

in reconstruction mode. Against a detector that reads notations the content clause cannot, prose mode leaked in 2,068 of 20,000 trials. That is the residual channel the paper names for prose mode, and it does not reach the mode we recommend. Removing any release-path defence except the domain clamp’s filter half restores the leak. Dropping that half instead puts a domain name the deployment does not have in front of the official. Four of the five availability defences start refusing honest traffic when removed, and the fifth releases every record and damages the structure of the document instead. The paper also states what it does not claim. The audit-signal plane is a design proposal rather than an evaluated capability. The distributed result is at pilot scale with one

peer per organisation rather than at government scale. The model result covers 4 open-weight models across 3 families in the parse position, with the clamp holding on each. The remaining gaps are size, redundancy and an independent review. The released code and runbook are built to support that work.

A Single-host Fabric pilot

This appendix keeps the earlier single-host comparison between the in-process pilot and a live four-organisation Fabric network. The pilot ran behind a gateway adapter presenting the same ledger interface. It exercised gRPC networking and TLS alongside endorsement and ordering as well as block commit and gossip dissemination. Table 10 sets the two backends side by side. A query takes a median of 488.1 ms end to end on Fabric, where network round trips dominate, against 2.22 ms of in-process orchestration. Committing a shared record takes a median of 1.10 s, bounded by the one-second block cut of the Raft ordering service.

The two columns are not a simultaneous measurement. The Fabric column was recorded before the defences of Section 7 and the in-process column after them. Those defences sit above the ledger interface and are backend independent, and the Fabric latencies would have risen as well. The in-process artifact records its recording time as 2026-08-17T03:46:16Z. The Fabric artifact predates that field, and its recording time is not recorded in the artifact. The table caption names each artifact by its content digest.

Table 10 Pilot in process versus on a single-host Fabric network. The two columns are separate recordings. Artifacts 20a9502be3e4 (Fabric) and 6f5d33b3e791 (in process).

Measure	In process	Live Fabric
Query latency, median	2.22 ms	488.1 ms
Query latency, p95	2.74 ms	542.0 ms
On-chain re-verify, median	0.13 ms	7.60 ms
Share commit, endorse to commit	n/a	1.10 s
Property tests passed	191 / 191	20 / 20
Out-of-scope releases	0	0

Figure 10 adds the worst-case fan-out curve for both backends. On Fabric, latency grows

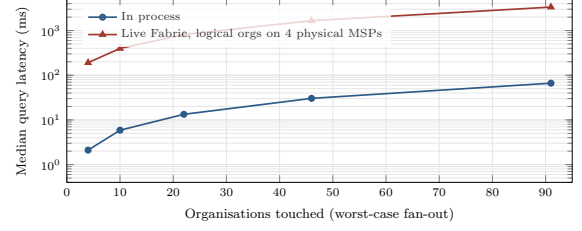


Fig. 10 Worst-case fan-out latency on a single host, in process and on Fabric. The two series are separate recordings.

from 191 ms to 3.35 s over the same range as Figure 8, with the released result and the zero-leak invariant unchanged. Its larger organisation counts are logical organisations mapped onto four physical membership service providers through an on-chain topology table. The curve therefore measures orchestration and gateway cost rather than endorsement fan-out across many physical peers. Each point is a median over 20 iterations with no dispersion reported, and it indicates where the cost sits rather than a scaling law.

Re-walking the chain does not detect a byte flipped in one peer’s block-store file. The walk stays intact because the peer re-syncs the block from the orderer and its peers on restart. Header-level linkage also survives corruption inside a transaction’s data region. The deterministic tamper result is the payload check of Section 11, which is what the verification property rests on.

References

- [1] M. Janssen, Y. Charalabidis, and A. Zuiderwijk, “Benefits, Adoption Barriers and Myths of Open Data and Open Government,” *Inf. Syst. Manag.*, vol. 29, no. 4, pp. 258–268, 2012, <https://doi.org/10.1080/10580530.2012.716740>
- [2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. Weed Cocco, and J. Yellick, “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains,” in *Proc. 13th EuroSys Conf.*, Porto, Portugal, 2018, art. 30, pp. 1–15, <https://doi.org/10.1145/3190508.3190538>
- [3] C. Ma, X. Kong, Q. Lan, and Z. Zhou, “The Privacy Protection Mechanism of Hyperledger Fabric and its Application in Supply Chain Finance,” *Cybersecurity*, vol. 2, no. 1, art. 5, 2019, <https://doi.org/10.1186/s42400-019-0022-2>
- [4] A. Barger, Y. Manevich, H. Meir, and Y. Tock, “A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric,” in *Proc. IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC)*, pp. 1–9, 2021, <https://doi.org/10.1109/ICBC51069.2021.9461099>

- [5] S. Ølnes, J. Ubacht, and M. Janssen, "Blockchain in Government: Benefits and Implications of Distributed Ledger Technology for Information Sharing," *Gov. Inf. Q.*, vol. 34, no. 3, pp. 355–364, 2017, <https://doi.org/10.1016/j.giq.2017.09.007>
- [6] V. L. Lemieux, "Trusting Records: Is Blockchain Technology the Answer?," *Rec. Manag. J.*, vol. 26, no. 2, pp. 110–139, 2016, <https://doi.org/10.1108/RMJ-12-2015-0042>
- [7] V. Thakur, M. N. Doja, Y. K. Dwivedi, T. Ahmad, and G. Khadanga, "Land Records on Blockchain for Implementation of Land Titling in India," *Int. J. Inf. Manage.*, vol. 52, art. 101940, 2020, <https://doi.org/10.1016/j.ijinfomgt.2019.04.013>
- [8] X. Zhao, S. Wang, Y. Zhang, and Y. Wang, "Attribute-based Access Control Scheme for Data Sharing in Hyperledger Fabric," *J. Inf. Secur. Appl.*, vol. 67, art. 103182, 2022, <https://doi.org/10.1016/j.jisa.2022.103182>
- [9] B. Otto, "A Federated Infrastructure for European Data Spaces," *Commun. ACM*, vol. 65, no. 4, pp. 44–45, 2022, <https://doi.org/10.1145/3512341>
- [10] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A Survey on Essential Components of a Self-Sovereign Identity," *Comput. Sci. Rev.*, vol. 30, pp. 80–86, 2018, <https://doi.org/10.1016/j.cosrev.2018.10.002>
- [11] Š. Čučko and M. Turkanović, "Decentralized and Self-Sovereign Identity: Systematic Mapping Study," *IEEE Access*, vol. 9, pp. 139009–139027, 2021, <https://doi.org/10.1109/ACCESS.2021.3117588>
- [12] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, and M. Conti, "A Survey on Decentralized Identifiers and Verifiable Credentials," *IEEE Commun. Surveys Tuts.*, vol. 27, no. 6, pp. 3641–3671, 2025, <https://doi.org/10.1109/COMST.2025.3543197>
- [13] E. Podda, P. Hölzner, A. Amard, J. Sedlmeir, and G. Fridgen, "The Impact of Zero-Knowledge Proofs on Data Minimisation Compliance of Digital Identity Wallets," *Internet Policy Rev.*, vol. 14, no. 3, 2025, <https://doi.org/10.14763/2025.3.2019>
- [14] M. Babel, L. Willburger, J. Lautenschlager, F. Völter, T. Guggenberger, M.-F. Körner, J. Sedlmeir, J. Strüker, and N. Urbach, "Self-Sovereign Identity and Digital Wallets," *Electron. Markets*, vol. 35, no. 1, art. 28, 2025, <https://doi.org/10.1007/s12525-025-00772-0>
- [15] D. Temoshok, D. Proud-Madruga, Y.-Y. Choong, R. Galuzzo, S. Gupta, C. LaSalle, N. Lefkowitz, and A. Regenscheid, "Digital Identity Guidelines," NIST Special Publication 800-63-4, Nat. Inst. Standards Technol., Gaithersburg, MD, Jul. 2025, <https://doi.org/10.6028/NIST.SP.800-63-4>
- [16] S. Rodriguez Garzon, A. Vaziry, E. M. Kuzu, D. E. Gehrmann, B. Varkan, A. Gaballa, and A. Küpper, "AI Agents with Decentralized Identifiers and Verifiable Credentials," in *Proc. 18th Int. Conf. Agents and Artificial Intelligence (ICAART)*, vol. 1, pp. 252–259, 2026, <https://doi.org/10.5220/0014234400004052>
- [17] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 9459–9474, 2020, <https://doi.org/10.48550/arXiv.2005.11401>
- [18] S. Zeng, J. Zhang, P. He, Y. Xing, Y. Liu, H. Xu, J. Ren, S. Wang, D. Yin, Y. Chang, and J. Tang, "The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG)," in *Findings of the ACL*, pp. 4505–4524, 2024, <https://doi.org/10.18653/v1/2024.findings-acl.267>
- [19] B. Chen, J. Tackman, M. Setälä, T. Poranen, and Z. Zhang, "Integrating Access Control with Retrieval-Augmented Generation: A Proof of Concept for Managing Sensitive Patient Profiles," in *Proc. 40th ACM/SIGAPP Symp. Appl. Comput. (SAC)*, pp. 915–919, 2025, <https://doi.org/10.1145/3672608.3707848>
- [20] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996, <https://doi.org/10.1109/2.485845>
- [21] P. Zhou, Y. Feng, and Z. Yang, "Provably Secure Retrieval-Augmented Generation," arXiv:2508.01084, 2025, <https://doi.org/10.48550/arXiv.2508.01084>
- [22] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," in *Proc. 16th ACM Workshop Artif. Intell. Secur. (AISec)*, pp. 79–90, 2023, <https://doi.org/10.1145/3605764.3623985>
- [23] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and Benchmarking Prompt Injection Attacks and Defenses," in *Proc. 33rd USENIX Security Symp.*, Philadelphia, PA, USA, 2024, pp. 1831–1847, <https://doi.org/10.48550/arXiv.2310.12815>
- [24] E. Debenedetti, I. Shumailov, T. Fan, J. Hayes, N. Carlini, D. Fabian, C. Kern, C. Shi, A. Terzis, and F. Tramèr, "Defeating Prompt Injections by Design," in *Proc. IEEE Conf. Secure and Trustworthy Machine Learning (SaTML)*, 2026, <https://doi.org/10.48550/arXiv.2503.18813>
- [25] L. Beurer-Kellner, B. Buesser, A.-M. Crețu, E. Debenedetti, D. Dobos, D. Fabian, M. Fischer, D. Froelicher, K. Grosse, D. Naef, E. Ozoani, A. Pavard, F. Tramèr, and V. Volhejn, "Design Patterns for Securing LLM Agents against Prompt Injections," arXiv:2506.08837, 2025, <https://doi.org/10.48550/arXiv.2506.08837>
- [26] T. Shi, J. He, Z. Wang, H. Li, L. Wu, W. Guo, and D. Song, "Progent: Securing AI Agents with Privilege Control," arXiv:2504.11703, 2025, <https://doi.org/10.48550/arXiv.2504.11703>
- [27] B. W. Lampson, "A Note on the Confinement Problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, 1973, <https://doi.org/10.1145/362375.362389>
- [28] M. Stonebraker and E. Wong, "Access Control in a Relational Data Base Management System by Query Modification," in *Proc. 1974 ACM Annual Conf.*, vol. 1, pp. 180–186, 1974, <https://doi.org/10.1145/800182.810400>
- [29] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy, "Extending Query Rewriting Techniques for Fine-Grained Access Control," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, pp. 551–562, 2004, <https://doi.org/10.1145/1007568.1007631>
- [30] Z. Zhang and A. O. Mendelzon, "Authorization Views and Conditional Query Containment," in *Proc. 10th Int. Conf. Database Theory (ICDT)*, LNCS vol. 3363, pp. 259–273, 2005, https://doi.org/10.1007/978-3-540-30570-5_18
- [31] M. Guarnieri and D. Basin, "Optimal Security-Aware Query Processing," *Proc. VLDB Endowment*, vol. 7, no. 12, pp. 1307–1318, 2014, <https://doi.org/10.14778/2732977.2733003>
- [32] D. E. Denning, P. J. Denning, and M. D. Schwartz, "The Tracker: A Threat to Statistical Database Security," *ACM Trans. Database Syst.*, vol. 4, no. 1, pp. 76–96, 1979, <https://doi.org/10.1145/320064.320069>
- [33] D. E. Denning, "Secure Statistical Databases with Random Sample Queries," *ACM Trans. Database Syst.*, vol. 5, no. 3, pp. 291–315, 1980, <https://doi.org/10.1145/320613.320616>
- [34] N. R. Adam and J. C. Worthmann, "Security-Control Methods for Statistical Databases: A Comparative Study," *ACM Comput. Surv.*, vol. 21, no. 4, pp. 515–556, 1989, <https://doi.org/10.1145/76894.76895>
- [35] D. P. Dobkin, A. K. Jones, and R. J. Lipton, "Secure Databases: Protection Against User Influence," *ACM Trans. Database Syst.*, vol. 4, no. 1, pp. 97–106, 1979, <https://doi.org/10.1145/320064.320068>
- [36] I. Dinur and K. Nissim, "Revealing Information While Preserving Privacy," in *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS)*, pp. 202–210, 2003, <https://doi.org/10.1145/1055558.1055570>

- 1145/773153.773173
- [37] K. Gai, Y. She, L. Zhu, K.-K. R. Choo, and Z. Wan, “A Blockchain-Based Access Control Scheme for Zero Trust Cross-Organizational Data Sharing,” *ACM Trans. Internet Technol.*, vol. 23, no. 3, art. 38, pp. 1–25, 2023, <https://doi.org/10.1145/3511899>
- [38] H. Ali, J. Ahmad, Z. Jaroucheh, P. Papadopoulos, N. Pitropakis, O. Lo, W. Abramson, and W. J. Buchanan, “Trusted Threat Intelligence Sharing in Practice and Performance Benchmarking through the Hyperledger Fabric Platform,” *Entropy*, vol. 24, no. 10, art. 1379, 2022, <https://doi.org/10.3390/e24101379>
- [39] H. Rashkin, V. Nikolaev, M. Lamm, L. Aroyo, M. Collins, D. Das, S. Petrov, G. Singh Tomar, I. Turc, and D. Reitter, “Measuring Attribution in Natural Language Generation Models,” *Comput. Linguist.*, vol. 49, no. 4, pp. 777–840, 2023, <https://doi.org/10.1162/coli.a.00486>
- [40] T. Gao, H. Yen, J. Yu, and D. Chen, “Enabling Large Language Models to Generate Text with Citations,” in *Proc. EMNLP*, pp. 6465–6488, 2023, <https://doi.org/10.18653/v1/2023.emnlp-main.398>
- [41] O. Honovich, R. Aharoni, J. Herzig, H. Taitelbaum, D. Kukliansky, V. Cohen, T. Scialom, I. Szpektor, A. Hassidim, and Y. Matias, “TRUE: Re-evaluating Factual Consistency Evaluation,” in *Proc. NAACL-HLT*, pp. 3905–3920, 2022, <https://doi.org/10.18653/v1/2022.naacl-main.287>
- [42] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, “FactScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation,” in *Proc. EMNLP*, pp. 12076–12100, 2023, <https://doi.org/10.18653/v1/2023.emnlp-main.741>
- [43] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A Large Annotated Corpus for Learning Natural Language Inference,” in *Proc. EMNLP*, pp. 632–642, 2015, <https://doi.org/10.18653/v1/D15-1075>
- [44] A. Williams, N. Nangia, and S. R. Bowman, “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference,” in *Proc. NAACL-HLT*, pp. 1112–1122, 2018, <https://doi.org/10.18653/v1/N18-1101>
- [45] P. He, J. Gao, and W. Chen, “DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing,” in *Proc. ICLR*, 2023, <https://doi.org/10.48550/arXiv.2111.09543>
- [46] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proc. EMNLP-IJCNLP*, pp. 3982–3992, 2019, <https://doi.org/10.18653/v1/D19-1410>
- [47] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, “Survey of Hallucination in Natural Language Generation,” *ACM Comput. Surv.*, vol. 55, no. 12, art. 248, pp. 1–38, 2023, <https://doi.org/10.1145/3571730>
- [48] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations,” in *Proc. Conf. Lang. Model. (COLM)*, 2024, <https://doi.org/10.48550/arXiv.2308.08155>
- [49] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, “MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework,” in *Proc. ICLR*, 2024, <https://doi.org/10.48550/arXiv.2308.00352>
- [50] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing Reasoning and Acting in Language Models,” in *Proc. ICLR*, 2023, <https://doi.org/10.48550/arXiv.2210.03629>
- [51] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language Agents with Verbal Reinforcement Learning,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 36, pp. 8634–8652, 2023, <https://doi.org/10.52202/075280-0377>
- [52] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating Noise to Sensitivity in Private Data Analysis,” in *Theory of Cryptography (TCC)*, LNCS vol. 3876, pp. 265–284, 2006, https://doi.org/10.1007/11681878_14

Statements and Declarations

Funding. The authors received no specific funding for this work.

Competing interests. The authors declare that they have no competing interests.

Author contributions. K. Almi’ani conceived the architecture, implemented the reference system and the experiment drivers, and wrote the manuscript. C. Zufferey contributed to the threat model and the security analysis. S. S. Kanhere shaped the security argument and the evaluation design through detailed technical review. N. Siyam contributed to the agent protocol design and the evaluation methodology. All authors reviewed and approved the final manuscript.

Data availability. The reference implementation and the live-Fabric adapter are publicly available at <https://github.com/papersC/puls>. So are the experiment drivers, every result artifact and the multi-host runbook. Two sets of *inputs* are not in the package, as Section 11 states. The single-domain cross-encoder row needs a model checkpoint the artifact environment cannot download, and the labelled 1,400-pair expanded grounding set is absent with its driver, so figures scored on that benchmark can be read from their artifact but not recomputed from source. Every measured value in this article is generated from those artifacts by the included scripts (`update_tex_numbers.py` and `gen_exp_numbers.py`) and substituted as a macro rather than edited by hand. The data figures are build artifacts as well. Figures 7, 8 and 10 read generated coordinates and are rebuilt after `gen_exp_numbers.py`. No third-party or personal data were used, and every organisation name, record and example here is synthetic.

Disclaimer. This article explores a generic reference architecture for confidential data sharing among government organisations. It is affiliated with no government or agency and describes no operational system.

Ethics approval. Not applicable. No human participants, animal subjects or personal data were involved.