ACaaS: Privacy-Preserving Access Control as a Service for Healthcare on the Cloud–Edge Continuum

Ha Xuan Son[1⋆], Phat T. Tran-Truong[2,3⋆], Bach Le-Xuan[2,3⋆⋆], Minh B. Nguyen[4], and Thanh Pham[5]

[1] Digital Economy, The Business School, SGS Campus, RMIT University Vietnam, Ho Chi Minh City, Vietnam
ha.son@rmit.edu.vn

[2] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam

[3] Vietnam National University Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam
phatttt@hcmut.edu.vn; lexuanbach@hcmut.edu.vn

[4] Can Tho University of Technology, Can Tho City, Vietnam
nbminh@ctuet.edu.vn

[5] School of Science, Engineering and Technology, SGS Campus, RMIT University Vietnam, Ho Chi Minh City, Vietnam
thanh.pham@rmit.edu.vn
[6]

Abstract. Enforcing consistent, privacy-aware access control across the healthcare cloud–edge continuum—a core service-composition challenge—is unresolved: centralized enforcement suffers WAN latency and outage fragility, while replicating full authorization engines at every edge exceeds device capabilities and offers no evidence that replicas agree. We propose **ACaaS**, an Access Control as a Service architecture that decomposes the XACML roles into microservices with typed, partition-tolerant RPC contracts, deploys a lightweight Policy Decision Point at the edge, minimizes records before they leave the device (pseudonymization, k-anonymity, role-driven redaction), and uses a permissioned ledger as a *synchronization substrate*—not merely an audit log—so that every edge provably enforces the policy version committed by cloud administrators. Five HIPAA/GDPR-derived security goals are discharged semi-formally (substitution-resistance lemma, five theorems, a knapsack-optimality placement argument, a closed-form reconciliation bound) and validated by twelve experiments, including an executable conformance suite mechanizing all theorems, a re-identification attack within one percentage point of the analytic $1/k$ bound, an ablation in which only the ledger detects tampered policy pushes, Byzantine fault injection, and a three-site partition/soak emulation sustaining 100% local availability with zero audit-log loss across 37 WAN partitions.

⋆ These authors contributed equally to this work.
⋆⋆ Corresponding author.

Keywords: Access Control · Cloud-Edge Continuum · Blockchain · Privacy · Microservices · Healthcare

1 Introduction

Modern healthcare delivery is being re-architected as a federation of *loosely coupled services* on a cloud–edge continuum: tertiary hospitals operate cloud back-ends hosting the electronic health record (EHR), while rural clinics, ambulances, and home-monitoring kits run on resource-constrained gateways such as Raspberry Pi-class boards [16,23]. This is a service-oriented computing problem: clinical workflows compose authentication, authorization, EHR retrieval, and audit services across domains with very different connectivity and trust profiles. Uniquely in healthcare, the *authorization* service must honor three constraints no existing design satisfies together: (i) the regulatory floor of HIPAA Security Rule §164.312(a)(1)–(d) and GDPR Articles 9, 25, and 32—role-restricted access, data minimization *by design*, tamper-evident audit trails on every protected health information (PHI) disclosure [21, 28]; (ii) availability during wide-area network (WAN) partitions, because rural clinics and mobile units routinely lose connectivity yet must keep ordering medications and triggering *break-glass* access; (iii) policy updates—formulary changes, revoked privileges, outbreak-driven consent overrides—must reach every site with cryptographic evidence of which version was enforced when.

Centralized cloud enforcement violates (ii): a Policy Decision Point (PDP) reachable only over the WAN adds latency and becomes a single point of failure [30,31]; replicating a heavyweight engine at every edge violates the continuum's resource asymmetry and offers no evidence that replicas agree [2]. Three literature gaps follow (elaborated in Sect. 2): the ledger in blockchain-based healthcare access control is an audit log or consent gate, not a *synchronization substrate* that lets each edge prove which policy version it was enforcing [4,13,14]; microservice access control targets the homogeneous cloud [5,11,26], leaving the *cross-tier decomposition* of the XACML roles under partition unexplored; and privacy preservation is rarely enforced where patient data originates [3,18,29].

This paper proposes **ACaaS**, a privacy-preserving *Access Control as a Service* for healthcare on the cloud–edge continuum: each XACML role is an independently deployable microservice with a well-defined RPC contract, composed across cloud, fog, and edge tiers under partial connectivity. Three design moves are domain-driven: (a) the compute profile of a rural gateway forces the edge PDP to be a lightweight WebAssembly engine covering the high-frequency clinical policy subset, with governance and break-glass review in the cloud; (b) for HIPAA §164.312(b) and GDPR Article 30 under clock skew and reordering, every policy version and access decision is anchored on a permissioned Hyperledger Fabric network with channels partitioned by read pattern and ACL; (c) for GDPR Article 25 (*data protection by design*), an edge-resident *Data Minimizer* runs *before* a PHI record may leave the device.

Our contributions are: **(i)** an ACaaS microservice architecture with explicit XACML role decomposition, tier mapping, and inter-service contracts under online and partitioned operation—squarely within the ICSOC themes of service composition and resilient service delivery on the cloud–edge continuum; **(ii)** an *adaptive policy deployment* strategy with a knapsack-style optimality argument (Proposition 1); **(iii)** a blockchain-backed *policy synchronization* protocol, distinct from audit logging, formalized through a substitution-resistance lemma, an eventual-consistency theorem, and a closed-form reconciliation bound, within a security analysis mapping the architecture to five HIPAA/GDPR-derived goals (G1–G5) under a Dolev–Yao adversary; and **(iv)** a twelve-experiment evaluation (E1–E12) pairing every security and privacy claim with a measured-vs-predicted outcome—an executable conformance suite, an empirical re-identification attack, a sync-substrate ablation, Byzantine fault injection, a three-site partition/soak emulation, and a real-Fabric sanity check.

The response letter and the supplementary material summarize how this revision addresses the ICSOC 2026 early-submission reviews.

2 Related Work

We position ACaaS against the three most directly comparable categories; orthogonal areas (TinyML biometric front-ends, general clinical decision support) are not surveyed, as they do not contribute to the access-control decision pipeline.

Microservice-based access control. Singh et al. [26], Gogineni and Mupparaju [11], and Alboqmi et al. [5] decompose the XACML PIP/PDP/PEP triad into independent services (the latter with risk-adaptive scoring)—but all on homogeneous clouds: none specifies an inter-service contract that survives a WAN partition between tiers, the canonical failure mode in multi-site healthcare. Policy-modeling and conflict-reconciliation work [8, 10, 15, 17] provides our ABAC underpinning but not tier placement.

Blockchain-anchored healthcare access control. Hybrid blockchain–ABAC frameworks [14, 22, 28] and smart-contract access control for IoT/smart healthcare [2, 4, 13, 19, 21, 30, 31] use the ledger as a tamper-evident audit log or patient-centric consent gate: it records what was decided, not which policy version was in force at the deciding peer. The closest design, QuickMedBlock [22], anchors policy hashes but separates no audit/sync read paths and has no offline reconciliation with monotonic versions.

Edge enforcement and privacy primitives. Pairing-free multi-authority CP-ABE [12] and dynamic edge integrity verification [16] bound what edge enforcement can afford. Homomorphic encryption with blockchain [3, 18, 29] and policy-hiding ABE [24] give stronger confidentiality at costs and composition risks [20] prohibitive on Pi-class devices; none runs minimization on the data-producing edge. Federated and post-quantum IAM [6, 9, 23, 25, 27] spans administrative boundaries but assumes always-connected clouds.

Positioning. Table 1 summarizes the comparison. ACaaS is, to our knowledge, the first design combining (a) a microservice decomposition of XACML across

Table 1: Closest related designs. "Sync substrate": the ledger anchors policy *version state*, not only audit; "minim.": applied before PHI leaves the device.

Approach	Micro-service	Edge PDP	Sync substrate	Edge-origin minim.	Offline enforce	Formal goals
Singh et al. [26]	✓	–	–	–	–	partial
Gogineni & Mupparaju [11]	✓	–	–	–	–	–
Alboqmi et al. [5]	✓	–	–	–	–	–
SMACS [19]	–	–	audit only	–	–	partial
Zhu et al. [31]	–	✓	audit only	–	–	–
Abid et al. [2]	–	–	audit only	–	–	–
QuickMedBlock [22]	–	–	hash only	–	–	–
Hao et al. [13]	–	–	audit only	–	–	partial
Kaur et al. [16]	–	✓	–	–	–	–
Rajagopal et al. [23]	partial	✓	–	–	–	–
ACaaS (this work)	✓	✓	✓	✓	✓	✓

cloud/fog/edge with a partition-tolerant contract, (b) a blockchain used as a *synchronization substrate* with provable monotonicity, separate from the audit channel, and (c) edge-origin minimization, in one healthcare-targeted architecture with a semi-formal security argument tied to HIPAA/GDPR goals. No individual primitive is novel; the *composition* provides properties (Lemma 1, Corollary 1, Theorem 3) that no prior system in the comparable set provides and that demonstrably disappear under component replacement (E10).

3 Preliminaries and Threat Model

3.1 Access-Control Model, Policy Objects, and Ledger Semantics

Let $\mathcal{S}$, $\mathcal{R}$, $\mathcal{A}$, and $\mathcal{E}$ denote finite sets of subjects, resources, actions, and environment conditions, each entity carrying an attribute vector $\mathrm{attr}(x) \in V_1 \times \cdots \times V_n$ (role, department, time-of-day, device posture). An *ABAC rule* is a function

$$\pi \;:\; \mathcal{S} \times \mathcal{R} \times \mathcal{A} \times \mathcal{E} \;\longrightarrow\; \{\textsc{Permit}, \textsc{Deny}, \mathrm{NA}\}, \tag{1}$$

with rule sets combined under *deny-overrides* [10, 11, 26]; RBAC is the special case in which $\mathrm{attr}(s)$ reduces to a role assignment, and risk-adaptive extensions (RAdAC) [5, 7] are permitted at the cloud-tier Policy Manager. The XACML reference architecture separates the Policy Administration, Decision, Enforcement, and Information Points (PAP, PDP, PEP, PIP) [10, 19]; in ACaaS each is an independent microservice mapped across tiers. A *committed policy* is a tuple $p = (id, \pi, v, h, \sigma)$ with $h = H(\pi)$ and $\sigma = \mathrm{Sign}_{sk}(id \,\|\, h \,\|\, v)$, where $id \in \mathrm{ID}$ is unique, π is the rule of (1) canonically encoded, $v \in \mathbb{N}$ a monotonically increasing version, h a SHA-256 digest, and σ the administrator's signature under an EUF-CMA-secure scheme. The ledger state is a partial function $L : \mathrm{ID} \to \mathbb{N} \times \{0,1\}^{256}$ mapping each committed identifier to its current $(version, hash)$; committing p is valid iff $id \notin \mathrm{dom}(L)$ or $v > L(id).version$, with σ verifying. The chaincode thus enforces

$$\mathsf{INV}_{\mathrm{mono}}\colon \quad \forall\, t_1 < t_2,\ \forall\, id \in \mathrm{dom}(L_{t_1}) \cap \mathrm{dom}(L_{t_2}) : \;\; L_{t_2}(id).v \;\geq\; L_{t_1}(id).v, \tag{2}$$

where L_t is the ledger projection at time t. Each edge e maintains a local store E_e with the same signature and watermark $w_e = \max\{v \mid \exists\, id : E_e(id).v = v\}$; the system is *consistent at* e iff

$$\forall\, id \in \mathrm{dom}(E_e) : \quad E_e(id) \;=\; L(id), \tag{3}$$

and *eventually consistent* iff a bounded T_{rec} after reconnection restores (3) on the subscription set; ACaaS instantiates these abstractions on Hyperledger Fabric [2, 22, 28, 31]. The synchronization latency for a batch of N updates is modeled as

$$T_{\mathrm{sync}}(N) \;\approx\; T_{\mathrm{order}} + T_{\mathrm{commit}} + T_{\mathrm{event}} + N \cdot (T_{\mathrm{fetch}} + T_{\mathrm{verify}}) + T_{\mathrm{ack}}, \tag{4}$$

where T_{order} and T_{commit} are the Raft and Fabric block-commit costs (constant in N within a block), T_{event} is the SyncChain push, and T_{fetch}, T_{verify} are per-policy; (4) predicts the linear-in-N regime observed in E3 and yields Corollary 1.

3.2 Privacy Notions at the Edge

Let D be an EHR dataset of records $r = (qi, sa)$, with $qi \in Q_1 \times \cdots \times Q_q$ as a vector of quasi-identifiers (birth date, zip code, gender, ethnicity) and sa the sensitive attribute. A *generalization* γ maps each Q_i to a broader domain. We say $\gamma(D)$ achieves k-*anonymity* [1, 18] if and only if

$$\forall\, r \in \gamma(D) : \quad \big|\{\, r' \in \gamma(D) \;:\; r'.qi = r.qi \,\}\big| \;\geq\; k. \tag{5}$$

Pseudonymization applies a deterministic mapping $\psi : \mathrm{ID} \times \mathcal{K} \to \{0,1\}^n$ to replace direct identifiers with unlinkable tokens. ACaaS implements $\psi(id, k) = \text{HMAC-SHA256}(k, id)$. This preserves referential integrity within a study but prevents cross-study linkage as long as the study key remains private.

Re-identification bound. Let $\mathcal{B}$ be an honest-but-curious clinician whose background knowledge fixes the quasi-identifier vector of one target patient. Given k-anonymous $\gamma(D)$ and a single released record, $\Pr[\text{link}] \leq 1/k$; under R *independent* releases from disjoint cohorts the cumulative success is $1-(1-1/k)^R$. Repeated releases over the *same* cohort enable intersection attacks, motivating the cohort-stability and release-frequency caps the Data Minimizer enforces (Sect. 4.3); both bounds are validated empirically in E9.

3.3 Threat Model and Security Goals

We target a multi-site healthcare consortium: the *cloud administrator* is trusted to author correct policies but subject to credential compromise; *edge devices* are semi-trusted (physically accessible, secrets in a secure element, bounded-duration disconnections); the *network* is controlled by a Dolev–Yao adversary $\mathcal{A}$ that may observe, delay, reorder, or drop messages but cannot forge signatures (EUF-CMA); *clinicians* are honest-but-curious, probing released records for inference. ACaaS pursues five goals, stated as predicates over executions: **G1 (Consistency)**: for every edge e and every id in its subscription set, $E_e(id) = L(id)$ holds eventually (3). **G2 (Auditability)**: every decision d at time t against version v has a signed LogChain entry (id, s, r, a, d, v, t). **G3 (Confidentiality)**:

only $(\psi(r.id),\ \gamma(r.qi),\ \varrho_{role}(r.sa))$ leaves the edge, with ϱ_{role} redacting fields outside the requester role's visible set and γ satisfying (5). **G4 (Availability)**: under connectivity loss of duration $T \leq T_{\max}$, the edge keeps deciding every policy in $\mathrm{dom}(E_e)$. **G5 (Compliance)**: LogChain and PolicyChain suffice for the access-trail and retention reports HIPAA and GDPR require [6, 10, 27, 28]. Section 4 derives the architecture directly from G1–G5.

4 The ACaaS Design

Four principles follow from G1–G5. **(P1) Separation of administration and enforcement**: authoring, validation, and conflict resolution stay in the cloud; only compiled policies reach the edge. **(P2) Tiered enforcement with authoritative fallback**: the edge decides the common case in milliseconds; the full cloud PDP is the fallback. **(P3) Blockchain as synchronization substrate, not only audit log**: the ledger anchors policy versions so each edge independently verifies what it enforces [2, 22, 28]. **(P4) Privacy enforced at the origin**: minimization runs at the edge—information *not released* cannot leak [1, 18].

4.1 Service Decomposition and Tier Mapping

ACaaS spans three tiers with distinct resource profiles, trust boundaries, and service responsibilities (Fig. 1, left; goal mapping in right column). The **cloud tier** centralizes policy governance: the Policy Manager Service (PAP) authors and validates rules, Conflict Resolution enforces deny-override semantics, and the full OPA PDP serves as the authoritative fallback—satisfying **P1** and anchoring **G5** through the PolicyChain and LogChain ledger channels. The optional **fog tier** offloads synchronization pressure from the WAN: a regional Policy Cache with TTL-based and version-driven invalidation reduces fetch latency for dense edge clusters, and Log Aggregation batches audit records before LogChain commit. The **edge tier**—a rural clinic gateway or mobile ambulance unit—enforces access locally via a lightweight OPA-WASM PDP against a hash-verified Local Policy Store, satisfying **G4** without WAN connectivity; the Data Minimizer enforces **G3** at the tier boundary so fog and cloud never receive a raw PHI record, and the Offline Queue preserves the **G2** audit trail across partitions. In summary, G1 and G5 are anchored at the cloud/ledger layer, G2 spans all tiers via the LogChain path, G3 is enforced exclusively at the edge before any record leaves the device, and G4 is a property of the edge tier alone; Sect. 4.5 formalizes these mappings and Sect. 5 validates each goal experimentally.

Inter-service contract. Tier interaction is defined by three RPC contracts, exposed as gRPC services. (C1) Authorization (PEP $\rightarrow$ PDP): the PEP submits the canonical request $q = (s, r, a, e)$; the responder returns a typed decision $d \in \{\textsc{Permit}, \textsc{Deny}, \mathrm{NA}\}$ with policy identifier id, enforced version v, and *obligations* $\mathcal{O}$ (e.g., a redaction map). The edge PDP responds first; on NA or unknown id, the PEP escalates to the cloud PDP via the API Gateway when

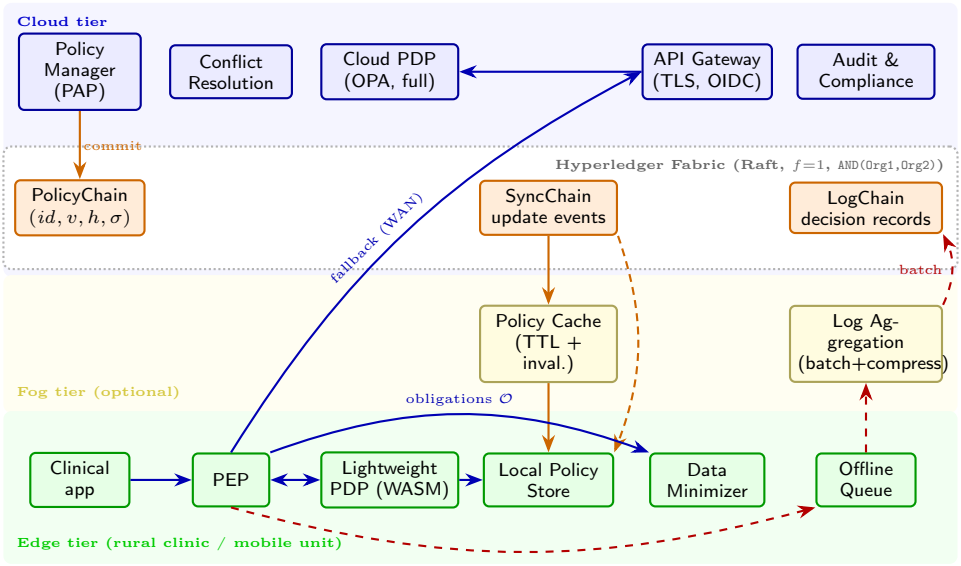


Fig. 1: The ACaaS architecture showing the service decomposition and data flows across the cloud-edge continuum. Authorization C1 (→), policy push C2 (→; no-fog path -➤), asynchronous audit push C3 (-➤).

the WAN is up, or fails closed. (C2) Policy push (PMS → edges): the PMS commits p on PolicyChain, publishes $\langle id, v_{\text{prev}}, v_{\text{new}} \rangle$ on SyncChain; edges pull the body over mTLS and accept iff $H(\pi)$ matches the ledger digest. (C3) Audit push (PEP → LogChain): every decision yields a signed record $\langle id, s, r, a, d, v, t, \text{seq}_e \rangle$; the monotonic per-edge seq_e lets Audit & Compliance detect gaps after a partition. The happy path is: (1) the PEP derives attr(s) from the Gateway's OIDC token and builds q; (2) the edge PDP evaluates the cached, version-pinned policy and returns $(d, id, v, \mathcal{O})$ in milliseconds; (3) the PEP applies $\mathcal{O}$ through the Data Minimizer; (4) the PEP asynchronously appends to LogChain via the Offline Queue—the only step that may block on the WAN. Three failure paths are explicit: (F1) *cache miss*—escalate to the cloud PDP, or fail-safe DENY for *safety-critical* policies if it too is unreachable; (F2) *WAN partition*—cache-and-buffer mode (Sect. 4.4); (F3) *hash mismatch on push*—reject the update, alert the PMS via SyncChain, keep the last hash-verified version, quarantine the id until re-commit.

4.2 Blockchain-Backed Policy Synchronization

The ledger utilizes three channels: *PolicyChain* stores (id, v, h, σ) tuples and enforces INV_{mono} of (2) in the endorsement chaincode; *LogChain* stores access decisions (only hash references or pseudonymized metadata on-chain, respecting erasure mandates); *SyncChain* carries the push events edges subscribe to. The separation is operational: read-heavy audit traffic cannot block governance writes, and ACLs are set per channel. Endorsement requires `AND(Org1.MSP,Org2.MSP)`,

so no single organization can unilaterally publish a version. Synchronization proceeds in three phases (step-numbered diagram in the supplementary material): **Phase A (cloud commit)**—the PMS canonicalizes π, increments v, commits (id, v, h, σ) on PolicyChain, and emits $\langle id, v_{\text{prev}}, v_{\text{new}} \rangle$ on SyncChain; **Phase B (online propagation)**—each subscribed edge fetches the body over mTLS, accepts iff $H(\pi)$ equals the ledger digest (mismatch triggers F3), atomically sets $E_e(id)$, and acknowledges; **Phase C (offline reconciliation)**—a reconnecting edge with watermark w_e invokes `getDelta`(w_e), verifies every hash against PolicyChain, applies the batch atomically, and drains its log buffer with seq_e-keyed deduplication [2, 28]. After Phase B/C, $E_e = L$ on the subscription set within T_{rec} (Theorem 1); substitution en route is detected except with negligible probability (Lemma 1).

4.3 Privacy-Preserving Data Flow at the Edge

The Data Minimizer is invoked unconditionally by the PEP between a PERMIT decision and the network response, so effective authorization is the conjunction of an access decision and a role-parameterized minimization transform: *Stage 1 (pseudonymization)* replaces direct identifiers with HMAC-SHA256(k_{study}, id) tokens under a rotated per-study key [1, 18]; *Stage 2 (k-anonymity)* generalizes retained quasi-identifiers until (5) holds ($k \in \{5, 10, 20\}$ evaluated); *Stage 3 (role-driven redaction)* replaces fields outside the $\mathcal{O}$-enumerated role-visible set with a fixed sentinel. Since the pipeline runs entirely on the edge, fog and cloud never observe a raw record (Theorem 3); E8 quantifies its cost.

4.4 Adaptive Deployment and Offline-First Operation

Policies are distributed across the continuum by a priority score

$$\text{score}(p) \;=\; \alpha \cdot f(p) \;+\; \beta \cdot c(p) \;+\; \gamma \cdot |p|^{-1}, \tag{6}$$

with $f(p)$ the observed access frequency, $c(p)$ the clinician-assigned criticality, and $|p|$ the compiled size, re-evaluated hourly. Under an edge cache hit-rate ρ_{edge}, the expected authorization latency is

$$\mathbb{E}[T_{\text{auth}}] \;=\; \rho_{\text{edge}}\, T_{\text{edge}} \;+\; (1 - \rho_{\text{edge}})\,(T_{\text{edge}} + T_{\text{WAN}} + T_{\text{cloud}}), \tag{7}$$

which (6) approximately minimizes (Proposition 1). During connectivity loss the edge moves from NORMAL to OFFLINE: only cached, hash-verified policies are enforced, unknown requests are default-denied (clinical fail-safe), and logs persist to the signed, monotonically numbered Offline Queue. On heartbeat restoration it enters RECONNECT-AND-SYNC, executes Phase C, drains the queue with seq_e deduplication, and returns to NORMAL; an edge offline longer than $T_{\max}$ enters read-only mode requiring administrator unlock, preventing indefinite divergence.

4.5 Security Analysis

We argue, semi-formally, that the architecture satisfies G1–G5, assuming EUF-CMA security of the signature scheme Σ, second-preimage resistance of H =

SHA-256, and Fabric Raft safety with fewer than f Byzantine orderers (f=1 of four here). Expanded proofs are in the supplementary material; every statement below is also mechanized as an executable test (E6) or measurement (E9–E11).

Lemma 1 (Substitution resistance of the sync substrate). *Let (id, v, h, σ) be committed on PolicyChain. The probability that an honest edge executing Phase B or C accepts a body $\pi' \neq \pi$ for this entry is at most $\mathbf{Adv}_H^{\mathrm{SPR}} + \mathbf{Adv}_\Sigma^{\mathrm{EUF\text{-}CMA}}$, which is negligible.*

Sketch. Acceptance requires $H(\pi') = h$—a second preimage—or altering the ledger entry, requiring a forgery of σ or violation of the `AND(Org1,Org2)` endorsement. This is a property of the *substrate*, not of transport encryption: plain pub/sub over TLS has no analogue, as E10 demonstrates. □

Theorem 1 (G1, eventual consistency). *For any edge e reconnecting at time t_0 with watermark w_e that remains connected for T_{rec} per (4) with $N = |\{id : L(id).v > w_e\}|$, the store E_e satisfies (3) at $t_0 + T_{\mathrm{rec}}$.*

Sketch. Phase C pulls exactly the policies with $v > w_e$ ($\mathsf{INV}_{\mathrm{mono}}$ forbids omissions); each body is accepted only under the hash check, so by Lemma 1 the installed body is the one the PMS signed, and atomic application rules out partial states. □

Corollary 1 (Reconciliation bound). *$T_{\mathrm{rec}}(N) \leq a + bN$ with $a = T_{\mathrm{order}} + T_{\mathrm{commit}} + T_{\mathrm{event}} + T_{\mathrm{ack}}$, $b = T_{\mathrm{fetch}} + T_{\mathrm{verify}}$; measured $a \approx 1.0$ s, $b \approx 13$ ms/policy (E3), so 200 missed updates reconcile in ≈ 3.6 s.*

Theorem 2 (G2, auditability). *Every decision returned by a PEP corresponds to exactly one LogChain entry $\ell = (id, s, r, a, d, v, t, \mathrm{seq}_e)$ that is either committed or pending in the Offline Queue; $\mathcal{A}$ cannot inject, alter, or suppress ℓ without detection.*

Sketch. ℓ is signed under the edge enrollment certificate before queueing; seq_e exposes gaps on ingestion; endorsement prevents a single peer from omitting committed entries. □

Theorem 3 (G3, edge-origin confidentiality). *For any PERMITed request, the tuple emitted by the edge to any non-edge tier is $(\psi(r.id), \gamma(r.qi), \varrho_{\mathrm{role}}(r.sa))$ with $\gamma(r.qi)$ satisfying (5); an honest-but-curious downstream observer re-identifies the target from a single release with probability at most $1/k$.*

Sketch. The Minimizer sits on every code path from raw record to network socket (invariant checked by E6); the bound is the k-anonymity guarantee under quasi-identifier-fixing background knowledge, and HMAC's PRF property blocks cross-release linking. □

Theorem 4 (G4, partition availability). *For any WAN partition of duration $T \leq T_{\max}$, the edge returns a decision for every $id \in \mathrm{dom}(E_e)$ within the local-evaluation budget.*

Theorem 5 (G5, compliance evidence). *From PolicyChain and LogChain alone, an auditor can reconstruct, for any decision d at time t, the exact policy text π and version v that produced d, and verify that π matches the digest committed at time $\leq t$.*

Sketch (Thms. 4, 5). Local evaluation touches only the hash-verified E_e; no step depends on cloud/fog reachability (import-level test, E6), and the Offline Queue is provisioned for $T_{\max}$. For Theorem 5, the LogChain entry carries (id, v), PolicyChain stores $H(\pi)$, the off-chain store retains π keyed by digest; recomputing $H(\pi)$ closes the loop. □

Proposition 1 (Near-optimality of the placement rule). *Let $\Delta = T_{\text{WAN}} + T_{\text{cloud}}$; pinning a policy set P_E with $\sum_{p \in P_E} |p| \leq M$ saves $S(P_E) = \sum_{p \in P_E} f(p)\,\Delta$ per request in expectation. Maximizing S is a 0–1 knapsack; greedy selection by density $f(p)\Delta/|p|$—the ranking that (6) induces with $\alpha \propto \Delta$, γ the size penalty, and $c(p)$ breaking ties—attains the fractional-relaxation optimum, is within one policy's saving of the integral optimum, and thus approximately minimizes $\mathbb{E}[T_{\text{auth}}]$ of (7).*

Sketch. Exchange argument (supplementary material); substituting the achieved hit-rate $\rho_{\text{edge}}(P_E) = \sum_{p \in P_E} f(p) / \sum_p f(p)$ into (7) gives the latency claim, which E7 validates at $R^2 = 0.998$. □

ACaaS defends against $\mathcal{A}$ in the network (Lemma 1), a single compromised edge (bounded scope, detectable via seq_e gaps), and an honest-but-curious downstream observer (Theorem 3); it does not defend against a compromised PMS administrator who legitimately commits a malicious policy (multi-org endorsement mitigates this).

5 Experiments and Results

ACaaS is evaluated on a cloud–edge testbed mirroring a geo-distributed clinical setting: the cloud tier hosts the Policy Manager, Fabric ledger, and full cloud PDP; each edge runs the lightweight PDP, Local Policy Store, Offline Queue, and Data Minimizer. The testbed runs a functionally equivalent Python prototype of the Rust production target, so absolute latencies are conservative upper bounds. Synthetic EHR workloads drive twelve experiments: E1–E5 measure performance, E6–E11 validate the security and privacy goals, E12 adds deployment-style evidence (Table 2). Full hardware specifications, workload parameters, baseline configurations, repetition counts, and variability measures for all twelve experiments are provided in the supplementary material (Sect. 2).

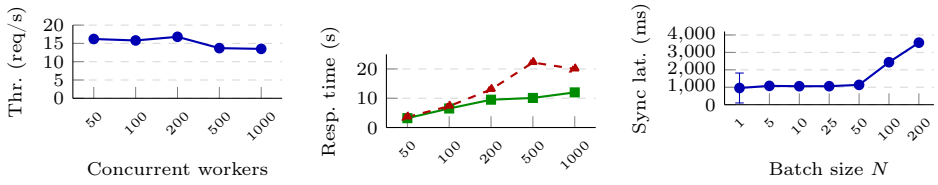
5.1 Performance (E1–E5)

Figure 2 reports load and synchronization behaviour; single-point comparisons are in Table 2, with E1/E4/E5 distribution plots in the supplementary material. **E1 (latency).** Removing the WAN round-trip cuts authorization latency

Table 2: Experiment-to-goal mapping and headline results.

Experiment	Goal(s) exercised	Headline result
E1: Cloud vs. edge latency	G4 (latency budget)	6.89 ms edge vs. 57.81 ms cloud
E2: Throughput under load	G4, G2 (audit pipeline)	0% failures up to 1,000 workers
E3: Sync consistency	G1 (Cor. 1)	1.06 s median to 10 endpoints
E4: Blockchain overhead	G2, G5 (compliance)	+3.22 ms vs. direct DB
E5: Offline resilience	G4 (partition avail.)	100% availability under WAN cut
E6: G1–G5 conformance suite	Theorems 1–5	21/21 assertions pass (54/54 total)
E7: Cache hit-rate vs. latency	Eq. (7), Prop. 1	$R^2 = 0.998$ vs. analytic model
E8: Minimizer per-stage cost	G3 (edge-origin minim.)	$\leq 18.4\,\mu$s per record
E9: Re-identification attack	Theorem 3 bound	within 1 pp of $1/k$ bound
E10: Ledger vs. pub/sub	Lemma 1 (substrate)	only ledger detects tampering
E11: Byzantine fault (f=1)	Theorem 1 safety/liveness	$\mathsf{INV}_{\text{mono}}$ holds; 100% liveness
E12: 3-site partition/soak (12 h)	G1, G2, G4 (deployment-style)	100% avail., 0 log loss, 37 partitions

from 57.81 ms (cloud, dominated by a 50 ms WAN delay) to 6.89 ms at the edge, keeping $\mathbb{E}[T_{\text{auth}}]$ of (7) within the clinical budget of G4 once $\rho_{\text{edge}} \geq 0.8$. **E2 (throughput).** Under 50–1,000 concurrent workers, throughput peaks at 16.8 req/s and degrades gracefully to 13.5 req/s with 0.0% failures (Fig. 2a); P95 response time grows sub-linearly until 500 workers, after which cloud audit-ingestion queue depth dominates (Fig. 2b)—the audit path (G2) holds without starving edge enforcement. **E3 (sync consistency).** Propagation after a global commit converges in median 1.06 s for a 10-policy batch and 3.56 s at 200 (Fig. 2c), matching the linear-in-N regime of (4); the fitted slope $b \approx 13$ ms/policy instantiates Corollary 1, closing the G1 loop. **E4 (ledger overhead).** Fabric anchoring costs +3.22 ms median over direct database writes (26.28 vs. 23.06 ms)—acceptable for the G2/G5 evidence it buys, especially off the synchronous PEP path. **E5 (offline resilience).** With the WAN severed, the edge sustained 100% availability on cached policies at 8.37 ms offline vs. 9.13 ms online; on restoration, queues drained without loss or reordering under seq_e deduplication—the operational expression of Theorem 4.



(a) Throughput (0% failures)

(b) Response time

(c) Sync latency (med. ± IQR)

Fig. 2: (a) Throughput and (b) response time (median —■—, P95 - -▲- -) vs. concurrency (E2); (c) sync latency vs. batch size (E3), linear in N as predicted by (4).

5.2 Quantitative Evidence for the Security and Privacy Goals (E6–E11)

The following experiments target the security and privacy claims directly; every statement of Sect. 4.5 has an executable counterpart.

E6 — G1–G5 conformance suite. Each theorem was translated into executable assertions and run as a pytest suite against the prototype—among them: eventual consistency after a 50-update offline burst, $\mathsf{INV}_{\text{mono}}$ on the chaincode, F3 hash-mismatch rejection, seq_e gap detection, drain-without-loss, the no-direct-identifier-egress invariant, the k-anonymity cohort check, the import-level no-WAN-coupling of the edge PDP, and an audit replay reconstructing (id, v, π) from the two chains. All 21 assertions pass; the full 54-test suite passes cleanly.

E7 — Cache hit-rate vs. authorization latency. A nine-setting sweep of ρ_{edge} (1,000 samples each) against (7) with measured $T_{\text{edge}} = 0.393$ ms, $T_{\text{WAN}} = 50$ ms, $T_{\text{cloud}} = 7.81$ ms coincides within ± 0.8 ms ($R^2 = 0.998$, Fig. 3a; 11.44 observed vs. 11.96 ms predicted at ρ_{edge}=0.8), validating the model behind Proposition 1.

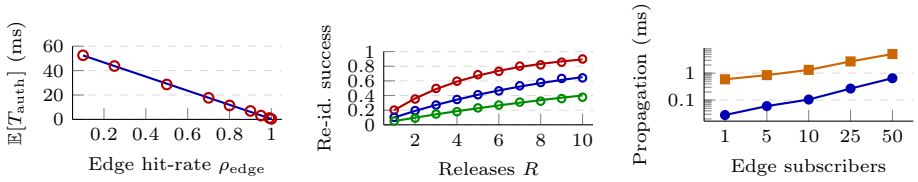
E8 — Data Minimizer per-stage cost (G3). Median per-record pipeline cost is 4.7/18.3/18.4 μs on small/medium/large records, dominated by HMAC pseudonymization (15.6 μs; plot in supplementary material); the cohort-level k-anonymity check scales linearly (16.2, 28.0, 53.8 μs for $k \in \{5, 10, 20\}$). Two orders of magnitude below the local decision budget, mandatory G3 enforcement on every PERMIT path is essentially free.

E9 — Empirical re-identification attack vs. the Theorem 3 bound. The quasi-identifier-fixing adversary of Sect. 3.2 ran on cohorts partitioned into anonymity classes (2,000 trials/setting): single-release success was 0.199, 0.109, 0.057 for $k \in \{5, 10, 20\}$—within one percentage point of $1/k$—and cumulative success over $R \in [1, 10]$ independent releases tracks $1-(1-1/k)^R$ almost exactly (Fig. 3b), discharging the privacy half of Theorem 3.

E10 — Sync-substrate ablation: ledger vs. pub/sub. The ledger costs 0.59 ms at one subscriber and 5.09 ms at 50; pub/sub costs 0.03 ms and 0.65 ms (Fig. 3c)—a 7.9× overhead but under 6 ms absolute. The pub/sub baseline silently propagates every tampered policy push (no body-to-hash anchoring), while the ledger catches every injection: exactly the separation predicted by Lemma 1.

E11 — Byzantine fault injection (f=1 of 4). In a four-peer endorsement ensemble with a two-signature threshold, one Byzantine peer ran passive (refuses to endorse), rewind (signs an outdated version), and fork (signs a different body) modes. Across 50 commits per scenario (200 total), liveness was 100% and $\mathsf{INV}_{\text{mono}}$ held with zero violations: honest peers always reach quorum, and the orderer rejects endorsements conflicting with the honest hash (fork) or violating monotonicity (rewind), confirming Theorem 1 under an active adversary.

E12 — Three-site partition/soak emulation. To bridge microbenchmarks and a future hospital deployment, we emulated twelve hours across three heterogeneous sites—urban hospital, district clinic, mobile unit—with site-specific



(a) E7: model vs. observed (b) E9: attack vs. bound (c) E10: substrate ablation

Fig. 3: Security and privacy evidence: (a) E7 cache-model validation (predicted ——, observed ○); (b) E9 re-identification vs. $1-(1-1/k)^R$ bound (k=5 ——, k=10 ——, k=20 ——); (c) E10 ledger —■— vs. pub/sub —●— (log scale)—only the ledger detects tampered pushes. E8/E11/E12 plots: supplement.

request rates and outage profiles: 27,360 requests, 29 policy updates, 37 WAN partitions. Every request was decided locally (100% availability, including 3,474 offline decisions); all 27,360 LogChain entries were ingested with zero loss; peak backlog was 240 and no site fell more than two updates behind (per-site table in supplementary material). 4.6% of decisions ran against a stale-but-hash-verified version and reconciled per Phase C—the price of availability that Theorem 4 accepts and Corollary 1 bounds. A controlled emulation, not a deployment—but it exercises the full offline-first pipeline end to end.

Real-Fabric sanity check. As E10–E11 use a Fabric-compatible in-memory mock for control, we deployed the official Fabric test network (two organizations, Raft) with a policy-hash chaincode: 8/8 commits (median 164 ms, P95 191 ms) and 8/8 reads (median 137 ms) succeeded—consistent with the $T_{\text{order}} + T_{\text{commit}}$ constants of (4), validating mock transferability.

Limitations. The edge PDP is a Python prototype (latencies are upper bounds for the Rust target); E12 uses synthetic outage distributions; E10–E11 run on the mock; and k-anonymity over small, churning cohorts remains exposed to composition attacks beyond the enforced caps—a local-differential-privacy layer and machine-checked proofs (TLA$^+$/ProVerif) of Phase C define the extended-version roadmap.

6 Conclusion

ACaaS composes the XACML roles as microservices across cloud, fog, and edge tiers under a partition-tolerant contract, with a ledger-based synchronization substrate and edge-origin minimization whose properties (Lemma 1, Theorems 1–5, Corollary 1) are established semi-formally and validated executably: across E1–E12, single-digit-millisecond edge authorization, tamper-evident synchronization that plain pub/sub provably lacks, Byzantine-resilient commits, and 100% offline availability with zero audit loss. Future work: the Rust edge engine, machine-checked Phase-C verification, differential privacy at the minimizer, and a partner-hospital shadow deployment.

Artefact availability and reproducibility statement. Source code, the G1–G5 conformance suite, and all benchmarks are at https://github.com/

SonHaXuan/ACaaS: `python -m benchmarks.run_all` regenerates every E6–E12 figure and JSON result, `pytest` runs the 54-test suite in seconds, and `docker-compose` brings up the full stack on one host. The supplementary material contains expanded proofs, the synchronization and Data Minimizer diagrams, E1/E4/E5 distribution plots, per-site E12 tables, and the real-Fabric transcript.

Acknowledgments

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

1. Aanjankumar, S., Muchahari, M.K., Urooj, S., Kaveri, P.R.: Enhanced consumer healthcare data protection through AI-driven TinyML and privacy-preserving techniques. IEEE Access (2025)
2. Abid, A., Cheikhrouhou, S., Kallel, S., Jmaiel, M.: A smart contract-based access control framework for smart healthcare systems. The Computer Journal (2024)
3. Aishwarya, B., Sriramya, P.: Decentralized and immutable record-keeping for data privacy-preserving in healthcare using homomorphic encryption in blockchain. In: 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (2025)
4. Al Amin, M., Altarawneh, A., Sarkar, S., Ray, I.: Blockchain smart contracts for policy compliance: A healthcare perspective. In: 2023 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC) (2023)
5. Alboqmi, R., Jahan, S., Gamble, R.F.: A risk adaptive access control model for the service mesh architecture. In: 2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI) (2024)
6. Alhuwayrini, A., Alqahtani, B., Almotairi, H., Saleem, K.: A unified security framework for addressing IAM and encryption challenges in hybrid eHealthcare cloud environments. In: 2025 IEEE International Conference on E-health Networking, Application and Services (Healthcom) (2025)
7. Ayedh M, A.T., Wahab, A.W.A., Idris, M.Y.I.: Enhanced adaptable and distributed access control decision making model based on machine learning for policy conflict resolution in BYOD environment. Applied Sciences (2023)
8. Centonze, P.: Security and privacy frameworks for access control big data systems. Computers, Materials and Continua (2019)
9. Demchenko, Y., Ngo, C., De Laat, C., Lee, C.: Federated access control in heterogeneous intercloud environment: Basic models and architecture patterns. In: Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E) (2014)
10. Farhadighalati, N., Estrada-Jimenez, L.A., Nikghadam-Hojjati, S., Barata, J.: A systematic review of access control models: Background, existing research, and challenges. IEEE Access (2025)
11. Gogineni, V., Mupparaju, S.: Fine-grained security in distributed systems: An abac implementation for microservice architectures. In: 2025 6th International Conference on Data Intelligence and Cognitive Informatics (ICDICI) (2025)

12. Guo, C., Peng, T., Zhang, J., Zhu, G.: A lightweight pairing-free multi-authority CP-ABE scheme for cloud-edge-assisted IoT. In: 2024 9th International Conference on Computer and Communication Systems (ICCCS) (2024)
13. Hao, L., Wang, R., Wang, X., Sajid, A.: Post-quantum-inspired scalable blockchain architecture for internet hospital systems with lightweight privacy-preserving access control. PLOS ONE (2025)
14. Kamala Kamatchi, G., Vanitha, K.: Analysis of blockchain-integrated access control models in distributed cloud environments. In: Proceedings of the 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA) (2025)
15. Kashmar, N., Adda, M., Atieh, M., Ibrahim, H.: SMART-AC: A new framework concept for modeling access control policy. Procedia Computer Science (2019)
16. Kaur, K., Rahman, S., Pal, S., Karmakar, C.: A patient-centric secure access control architecture with dynamic edge data integrity verification in internet of medical things. Ad Hoc Networks (2026)
17. Kuang, T.P., Ibrahim, H.: Security privacy access control for policy integration and conflict reconciliation in health care organizations collaborations. In: iiWAS2009 - The 11th International Conference on Information Integration and Web-based Applications and Services (2009)
18. Lakshmanan, M.: A secure and privacy-preserving framework for real-time data streams in telemedicine. In: 2025 8th International Conference on Circuit, Power and Computing Technologies (ICCPCT) (2025)
19. Liu, B., Sun, S., Szalachowski, P.: SMACS: Smart contract access control service. In: Proceedings of the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2020)
20. Mainardi, N., Barenghi, A., Pelosi, G.: Plaintext recovery attacks against linearly decryptable fully homomorphic encryption schemes. Computers & Security (2019)
21. Musa, N.S., Murugan, T., N, N.A., Mirza, N.M.: Research study on securing the cloud: Utilizing conventional and blockchain-based access control mechanisms. Journal of Cloud Computing (2025)
22. Punia, A., Gulia, P., Gill, N.S., Baqasah, A.M.: QuickMedBlock: A framework for enhanced attribute-based access control using blockchain for EHR in cloud. Peer-to-Peer Networking and Applications (2025)
23. Rajagopal, S.M., M., S., Buyya, R.: Leveraging blockchain and federated learning in edge-fog-cloud computing environments for intelligent decision-making with ECG data in IoT. Journal of Network and Computer Applications (2025)
24. Ruan, C., Hu, C., Zhao, R., Yu, J.: A policy-hiding attribute-based access control scheme in decentralized trust management. IEEE Internet of Things Journal (2023)
25. Sette, I.S., Chadwick, D.W., Ferraz, C.A.G.: Authorization policy federation in heterogeneous multicloud environments. IEEE Cloud Computing (2017)
26. Singh, A., Raj, V., Ravichandra, S.: Integration of attribute-based access control in microservices architecture. In: Lecture Notes in Networks and Systems. Springer (2022)
27. Srujana, K., Joginpelly, S.K., Ramachandra, A.C., Shanmuga Sundaram, K.: PQ-FIM-PCES: Enhancing privacy compliance in global cloud systems using post-quantum cryptography and federated identity management. In: International Conference on Software, Systems and Information Technology (SSITCON) (2025)
28. Sunitha, B.J., Kumar, S.S.: A novel hybrid blockchain-abac framework for multi-layered access control in cloud-based healthcare systems: Performance optimization and regulatory compliance. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications **16**(3), 178–197 (2025)

29. Verma, A.K., Patel, P., Bhatia, V.S., Ranjan, P.: Secure federated multi-party computation with homomorphic encryption and blockchain for preserving privacy in cloud storage of medical records. In: 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (2025)
30. Xu, R., Chen, Y., Blasch, E.: Decentralized access control for IoT based on blockchain and smart contract. In: Modeling and Design of Secure Internet of Things (2020)
31. Zhu, Y., Huang, C., Hu, Z., Al-Dhelaan, M.: Blockchain-enabled access management system for edge computing. Electronics **10** (2021)