International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 415-432
https://doi.org/10.70917/ijcisim-2026-5317
Article

Minimum-Exposure Observability: Task-Specific Diagnostic Utility Under Distributed-Trace Minimization

Lokeswari Vejju

Independent Researcher, Seattle, Washington, USA
Email: vejjulokeswari@gmail.com

Abstract: Distributed traces support diagnosis but can expose sensitive identifiers, payload content, and business context, motivating telemetry minimization. Prior privacy-preserving telemetry work quantifies utility, but does not generally organize application-trace evaluation around whether the same minimized representation preserves different diagnostic tasks differently. We therefore ask how privacy-driven minimization affects task-specific diagnostic utility and whether different diagnostic tasks require different minimum telemetry representations. We evaluate five minimization treatments — suppression, pseudonymization, generalization, whole-trace sampling, and a composite treatment — using a deterministic five-service in-process simulator and six independently defined and independently validated diagnostic tasks. Under the composite treatment, which suppresses actor identity while generalizing payload and business context, cross-request incident-cohort attribution was indeterminate in all 500 confirmatory windows and five fault strata, while context-associated failure diagnosis remained fully preserved over the same windows and strata. Through a different minimization mechanism within the same confirmatory design, whole-trace sampling at $p = 0.50$ produced a statistically significant task-specific difference in operational affected-set-F1 loss between these tasks ($D = -0.0396$, 95% CI $[-0.0632, -0.0161]$, $p = 0.00110$). These results support a task-dependent view of diagnostic sufficiency. We do not identify a globally minimal trace schema; rather, utility preservation must be evaluated with respect to the diagnostic capability being preserved.

Keywords: Distributed tracing; telemetry minimization; privacy-preserving observability; diagnostic utility; distributed systems; trace sampling

1. Introduction

1.1 Motivation / Problem Context

Distributed tracing has become a standard instrument for understanding the behavior of microservice-based systems: a trace records the causal path a request takes through a set of services, along with the timing, status, and contextual attributes attached to each step. This visibility is what lets an engineer go from "a request failed" to "this specific service, under this specific condition, produced this specific failure." OpenTelemetry [1], a widely adopted observability framework and specification, encodes this expectation directly into its data model: traces exist to preserve causal structure, and additional contextual information can enable diagnostic questions that coarser telemetry cannot answer.

The same richness that makes a trace diagnostically useful can also make it a liability. A trace that faithfully records which user issued a request, what resource they touched, and the exact parameters involved is, by construction, also a record of that user's activity. Where a trace is exported to a third-party analytics platform, an external support team, or a long-retention audit store, that combination of diagnostic value and personal or business-sensitive content is not incidental — it is the same information serving two different purposes at once. OpenTelemetry's own guidance

on handling sensitive data [2] acknowledges this directly, while leaving the actual decision of what to redact, hash, or generalize to the implementer.

The natural response is some form of data minimization: strip, pseudonymize, or generalize the fields that carry sensitive content before a trace leaves the boundary where it can be trusted. This is not a new idea, and we do not present it as one. Redaction, hashing, and field-level generalization are established techniques, and OpenTelemetry itself ships tooling for exactly this purpose. What is less settled is a more basic question: minimized *relative to what*? Prior privacy-preserving telemetry work quantifies utility, but does not generally organize application-trace evaluation around whether the same minimized representation preserves different diagnostic tasks differently. A trace that lets an engineer identify which service failed is not necessarily a trace that lets an analyst determine which customer was affected by a given incident, and a trace stripped down to satisfy the first need may or may not satisfy the second. Treating diagnostic utility as one property, rather than a set of distinct, task-specific capabilities, risks either overestimating what a minimized representation actually preserves, or discarding information that specific diagnostic tasks genuinely require while gaining no privacy benefit relevant to those tasks.

This motivates the question we study:

How does privacy-driven minimization of distributed-trace data affect task-specific diagnostic utility, and do different diagnostic tasks require different minimum telemetry representations?

We investigate this question empirically, using distributed traces as the object of study and a set of independently defined, independently validated diagnostic tasks as the instrument for measuring utility under five minimization treatments — suppression, pseudonymization, generalization, sampling, and a composite treatment combining several of these. Four of the tasks rely primarily on structural, status, timing, and other non-content operational metadata that content-oriented minimization largely leaves intact. Consistent with those dependencies, all four remain fully preserved under the field-level content treatments we evaluate — suppression, pseudonymization, generalization, and the composite treatment; this is a real result, not a null one, and we report it as such rather than treating it as an experimental shortfall.

The two remaining tasks are batch-level: they evaluate diagnostic capabilities whose information requirements intersect directly with the classes minimization actually targets — persistent actor correlation in one case, and contextual resource information in the other — and both were defined and validated on unminimized data before any treatment was ever evaluated against them. Under those conditions, the same minimization treatment produces markedly different outcomes depending on which task is asked of the resulting trace, including one composite treatment that fully preserves one task while rendering the other indeterminate, with the same divergence observed across all 500 confirmatory windows and all five fault strata. Through a different minimization mechanism within the same confirmatory design, we further confirm that this task-dependence is not limited to cases where a treatment simply removes a field a task happens to need outright: even a volume-reducing mechanism that never alters the content of any individual record produces a statistically distinguishable difference in how much diagnostic completeness two different tasks lose.

Taken together, these results show that diagnostic sufficiency is task-dependent: a telemetry representation that preserves one diagnostic capability may fail to preserve another, and utility preservation cannot be treated as a task-independent property of a minimized trace. Our experiments do not attempt to identify a globally minimal trace schema, nor do they exhaustively search the space of possible telemetry subsets for each task; rather, they show that claims of preserved diagnostic utility must be made with respect to the specific diagnostic task being performed. We observe task-specific sufficiency requirements under the representations we evaluate; we do not claim to enumerate or prove the globally minimal representation for any task. We discuss the boundaries of this scope directly in our limitations.

1.2 Contributions

This paper makes three contributions.

A task-based evaluation methodology. We develop and evaluate a task-based methodology for measuring diagnostic utility under telemetry minimization as a set of independently defined, independently validated tasks rather than a single aggregate score, and we apply this methodology to distributed traces specifically, evaluated using minimization techniques established in prior work — suppression, pseudonymization, generalization, and sampling — with one treatment combining several of these into a composite defined for this study, rather than inventing a wholly new minimization approach for the purpose. Each task is validated in isolation, on unminimized data, before any minimization treatment is applied to it — a discipline intended to keep the measurement instrument itself separate from the experimental question it is used to answer.

An empirical characterization across five minimization treatments. We evaluate suppression, pseudonymization, generalization, sampling, and a composite treatment against six independently validated diagnostic tasks, covering failure-service identification, request-path reconstruction, failure-class discrimination, latency-root-cause localization, cross-request actor-cohort attribution, and context-associated failure diagnosis. This gives a systematic account of which specific telemetry classes each task actually depends on, rather than a single before/after utility comparison.

Evidence that diagnostic sufficiency is task-dependent. We show, first, that a single minimization treatment can fully preserve one diagnostic task while rendering another indeterminate, with the same divergence observed across all 500 confirmatory windows and all five fault strata. We show, second, that this task-dependence extends beyond cases of outright field removal: a volume-reducing treatment that never alters retained content still produces a statistically significant difference, within the same confirmatory design, in how much diagnostic completeness two distinct tasks lose. Together, these results support treating telemetry sufficiency as dependent on the diagnostic task being performed, rather than as a task-independent property of the trace.

1.3 Related Work

1.3.1 Privacy-Preserving Logging and Telemetry

Proteus (PoPETs 2026) [3] targets privacy-preserving device logs, using keyed-hash pseudonymization combined with rotating encryption to preserve correlation, linkage, and timeline-reconstruction capability across log entries while withholding plaintext PII from an honest-but-curious analytics backend. Proteus demonstrates that privacy-preserving derivatives can retain correlation and timeline-reconstruction capabilities without exposing plaintext PII; its evaluation emphasizes security guarantees and systems overhead rather than task-differentiated downstream diagnostic accuracy, and its object of study is mobile device logs rather than distributed application traces.

Zhou et al. (SSDBM 2023) [4] study redaction quality directly, using source-code data-flow analysis to reduce both false-positive (unnecessarily redacted, potentially reducing diagnostic utility) and false-negative (sensitive information retained) outcomes in diagnostic log redaction. The paper explicitly recognizes that removing sensitive diagnostic information can reduce what a diagnosis needs — the same tension this paper studies — but its evaluation is centered on redaction accuracy, rather than success on multiple downstream diagnostic tasks.

Where Proteus and Zhou et al. focus on protecting log content, a parallel line of work quantifies privacy-utility tradeoffs in network telemetry directly. Mohammady et al. (CCS 2018) [5] established that such tradeoffs can be rigorously, empirically quantified, using multi-view anonymization validated against real ISP traffic. This work is best understood as a methodological ancestor: it demonstrates that "measure the tradeoff, don't just assert it" is achievable for trace-like data, but its object of study is network traces, rather than application-level distributed traces, and its utility analysis is not organized around distinct downstream diagnostic tasks. PrvTel [6] (NSDI 2026) extends this line toward learned representations, retaining NetFlow and cloud telemetry under differential privacy while maximizing downstream query accuracy through a domain-specialized generative model with structure-aware noise. PrvTel establishes that DP-based generative retention can substantially improve query accuracy over prior DP baselines for network-level telemetry; its utility measure is aggregate query accuracy rather than a set of distinct

diagnostic tasks, and its telemetry object remains network/cloud-level flows rather than application-level distributed traces.

1.3.2 Trace Reduction and Diagnostic Utility

Autoscope (2025) [7] reduces distributed-trace storage volume through code-aware, span-level sampling, and directly evaluates the resulting traces' downstream usefulness: across two open-source microservice benchmarks, it reduces trace size by 81.2% while maintaining 98.1% faulty-span coverage, and improves MRR across four RCA methods — TraceRCA, TraceAnomaly, TraceContrast, and MicroRank — by 8.3% on average. Autoscope directly demonstrates that trace reduction can be evaluated against downstream diagnostic performance; our study asks a different question — how minimization motivated by privacy, rather than storage cost, changes the sufficiency of telemetry for different tasks. Autoscope's optimization target is storage volume, and it has no privacy or sensitive-content axis comparable to the exposure metrics we report alongside utility.

1.3.3 Diagnostic Evaluation and Trace Analysis

RCAEval (WWW'25) [8] establishes explicit, benchmarked diagnostic outcomes: three datasets comprising 735 failure cases collected from three microservice systems, and a reproducible evaluation framework spanning fifteen baseline RCA methods across coarse- and fine-grained localization. Its role for this paper is methodological rather than competitive — it reinforces that diagnostic utility should be measured as explicit, scored task performance rather than assumed from telemetry availability, the same discipline this paper applies to its own task solvers. RCAEval does not manipulate telemetry according to any privacy or minimization treatment; utility and privacy are entirely orthogonal in its design. Contrast (2026) [9] complements that perspective from a different angle, showing that diagnostically useful differences between distributed traces can manifest across structural, temporal, critical-path, and semantic dimensions — support for treating trace utility as multidimensional, though privacy minimization is outside its problem entirely.

1.3.4 Information Sufficiency Under Privacy Transformations

Say Something Else (2026) [10] studies a structurally related question in a different domain: for LLM-mediated conversational privacy, does strategy choice (suppression, generalization, or pseudonymization) determine how much private information is protected while task utility is preserved? Their large-scale evaluation across 792 scenarios and seven frontier models finds that scenario context accounts for the overwhelming majority of outcome variance (98.4%), against a comparatively small — though statistically significant — contribution from strategy choice (1.0%) and negligible contribution from model identity (0.2%); generalization is, notably, the only strategy their study finds Pareto-dominated by no protection at all. Their work shows that context strongly modulates privacy-strategy effectiveness in conversational LLM communication. Our work examines a different interaction: whether the same minimized distributed-trace representation supports different diagnostic tasks differently. The object of study, the mechanism space, and the question being asked are each distinct, though both papers share a discipline of measuring privacy-utility tradeoffs empirically rather than assuming them.

1.3.5 Synthesis

Existing work separately establishes privacy-preserving telemetry mechanisms, privacy-utility tradeoffs for network- and log-level data, trace-reduction methods that retain diagnostic value, and rigorous, standardized evaluation of diagnostic systems. Across these lines of work, a less characterized dimension is whether utility retained after distributed-trace minimization is task-dependent: whether the same minimized representation can remain sufficient for one diagnostic capability while becoming insufficient for another. Our study evaluates this interaction directly.

2. Materials and Methods

2.1 Study Design and Research Question

This paper investigates: *how does privacy-driven minimization of distributed-trace data affect task-specific diagnostic utility, and do different diagnostic tasks require different minimum telemetry representations?*

We answer this empirically using one synthetic distributed system, six independently defined and independently validated diagnostic tasks, and six telemetry conditions — one full-telemetry control (M0) and five minimization treatments (M1–M5). Each task is scored against task-specific ground truth maintained outside the telemetry itself and never exposed to any solver. Diagnostic utility and privacy/exposure are measured as two separate, independently defined families of metrics (Sections 2.6 and 2.7); nothing in the utility metrics feeds into the exposure metrics or vice versa. Where a pairing is defined — individual traces for M0–M5 applied to Tasks 1–4, or batch windows for Tasks 5–6 — every treatment is derived from the same underlying base trace(s), so treatment effects are measured as within-unit differences rather than across independently sampled data. For Tasks 5 and 6, the window — not the individual trace — is the unit of inference throughout this study, including in the statistical analysis (Section 2.10). Stage-2 confirmatory statistical procedures are described separately, in Section 2.10.

2.2 Synthetic Distributed System

All experiments run against one independently designed, fully synthetic five-service topology:

Client -> API Gateway -> Request Service -> Resource Service -> Fulfillment Service -> Notification Service

We state this plainly rather than leaving it implicit: **this is an in-process deterministic simulator, not five deployed microservices.** Each "service" is a function-level step in a single process, driven by a virtual clock rather than wall-clock time, so that span timing — including deliberately overlapping parent/child intervals used to exercise self-time computation — is exactly reproducible from a given seed rather than subject to OS scheduling variance.

Synthetic actors and resource values are drawn from fixed, independently constructed pools, not derived from any production system, production dataset, or employer-specific architecture. Five fault classes are injected at defined points in the call chain: dependency timeout, latency regression, authorization failure, retry amplification, and partial transaction failure. Ground truth — which service is responsible for a fault, what the true call-graph structure is, which span carries the injected excess latency, which actor or resource category is associated with an incident — is computed by the simulator at trial-construction time and stored entirely outside the telemetry structure any solver receives; ground truth is consulted only by the scoring layer.

We chose a synthetic system for three reasons: it gives us fully controllable, unambiguous ground truth for every task; it makes executions deterministic from the recorded trial seeds together with the frozen experiment-level configuration — M2 additionally uses the fixed run-level pseudonymization key described in Section 2.4, so a seed alone is not sufficient to regenerate the exact pseudonym values without that key; and it keeps the study fully independent of any employer system, production architecture, or non-public technical information under the study's documented provenance controls.

2.3 Threat Model and Sensitive-Data Classes

We assume an honest-but-curious telemetry consumer: a backend or downstream analyst that legitimately performs diagnostic analysis over exported traces but should not receive unnecessary plaintext identifiers, sensitive payload content, or unnecessary cross-request actor linkability. The source application and the minimization component itself are trusted. Cryptographic compromise, timing side channels, endpoint compromise, and malicious instrumentation are outside this threat model's scope.

Every telemetry field is assigned to one of six classes:

- **A — Direct identifiers**: user_id, session_id.
- **B — Correlation identifiers**: trace_id, span_id, parent_span_id, case_id.
- **C — Payload/business context**: request payload, error detail text, resource amount.
- **D — Structural/topology metadata**: service name, span name, span kind.

- **E — Timing**: start time, end time.
- **F — Status/operational behavior**: status code, exception type, retry count.

Class E is deliberately **outside** the threat model above — timing is treated as non-sensitive throughout this study. Because timing is outside the threat model, the frozen field-level minimization treatments do not transform Class E; an early implementation draft that coarsened timestamps under generalization was corrected before any experimental data existed (Section 2.8).

2.4 Telemetry Minimization Treatments

Treatment	Operation
M0	Full telemetry (control)
M1	Suppression of Class A and Class C
M2	Deterministic pseudonymization of Class A and Class B
M3	Generalization of Class C
M4	Whole-trace Bernoulli sampling, p = 0.50
M5	Composite: suppress Class A, pseudonymize Class B, generalize Class C, retain D/E/F

M2's pseudonymization is implemented as a domain-separated HMAC over each identifier, keyed with a single secret fixed for the entire experimental run (never derived per-trial), so that the same underlying identifier always maps to the same token across independently generated traces. Output lengths follow the OpenTelemetry [1] TraceId (16-byte) and SpanId (8-byte) encoding so that pseudonymized traces remain structurally valid traces. span_id and parent_span_id are pseudonymized under the same domain, so that a token's parent-child relationship is preserved exactly: a span's pseudonymized ID is identical wherever it appears, including as some other span's parent reference. Note precisely what this treatment does and does not do: **M2 pseudonymizes and thereby preserves cross-trace actor linkage** (Section 2.7, E3); **M5 suppresses Class A outright**, and, having no actor-identity representation left to pseudonymize, does not preserve actor linkage — M5's own pseudonymization applies only to Class B (trace/span/case identifiers), which is a structural property, not an actor-identity property. All five treatments operate on independent copies of the same base trace(s), never mutating the original, so the M0 condition remains an untouched reference for every paired comparison.

2.5 Diagnostic Tasks

Six tasks, independently defined and independently validated before any treatment evaluation (Section 2.8). Tasks 1–4 operate at the individual-trace level; Tasks 5–6 operate at the level of a multi-trace batch window, and the window — not the individual trace — is the unit of inference for those two tasks throughout this study (Section 2.10).

T1 — Failing-service identification. Output: the service responsible for the injected fault. When one or more spans are marked ERROR, the solver selects the deepest error span in the trace hierarchy; otherwise, it selects the span with maximum self-time rather than maximum raw duration.

T2 — Request-path reconstruction. Output: the reconstructed parent→child span-edge set, scored by edge-set F1 against the true call graph.

T3 — Failure-class discrimination. Output: one of the five injected fault classes, using status, exception-type, retry-count, and timing signals visible in the trace.

T4 — Latency-root-cause localization. Output: the span responsible for excess self-time (duration minus the union of child intervals, correctly handling overlapping children). Scored only on the three fault classes that are latency-relevant (dependency timeout, latency regression, retry amplification); authorization failure and partial transaction failure are not latency problems by construction and are excluded from this task.

T5 — Cross-request incident-cohort attribution. Window-based. Given a batch of traces, some sharing a synthetic actor identity and some not, identify which actor cohort is disproportionately associated with a target fault signature, and which specific traces in the batch are affected. This task's identification depends on a surviving representation of Class A actor identity — plaintext (M0/M3/M4) or a consistent pseudonym (M2) both suffice, since attribution is judged by underlying cohort membership rather than surface token identity; where Class A is removed entirely (M1, M5), no representation of the required signal survives and the task is indeterminate (Section 2.6).

T5 solver. The target fault class is supplied as the diagnostic query. The frozen T3 classifier identifies traces exhibiting that fault, traces are grouped by their visible actor representation, and the solver selects the group with the highest target-fault concentration. The predicted affected set consists of target-fault traces within that selected group. Ground-truth affected traces are those that both belong to the designated incident actor and carry the target fault.

T6 — Context-associated failure diagnosis. Window-based. Given a batch of traces spanning three resource-amount bins (the same low/mid/high bins used by the M3 generalization treatment — not a separately invented category scheme), identify which bin is disproportionately associated with a target fault signature, and which specific traces are affected. This task depends on a surviving representation of Class C resource-amount information — exact (M0/M2/M4) or generalized-to-bin (M3, and the generalization component of M5) both suffice; where Class C is removed entirely (M1), the task is indeterminate.

T6 solver. The same target-fault classification procedure is used, but traces are grouped by resource category. Exact resource amounts are mapped through the same frozen low/mid/high binning function used by M3; already-generalized values are consumed directly. The solver selects the category with the highest target-fault concentration, and the affected set comprises target-fault traces in that category. This shared binning function is what guarantees exact (M0/M2 and retained M4 traces) and generalized (M3/M5) resource-amount representations are interpreted on identical category boundaries, rather than risking a mismatch between how M3 generalizes values and how T6 interprets them.

Batch construction for T5/T6. Each window contains 48 traces, constructed with a fixed, non-probabilistic concentration imbalance rather than independent random assignment, so that task difficulty is not incidentally determined by sampling luck: for T5, six actors × eight traces each, with the incident actor's trials at a 6-of-8 (75%) target-fault concentration and every other actor's trials at a 1-of-8 (12.5%) background concentration; for T6, three resource-amount bins × sixteen traces each, with the incident bin at a 12-of-16 (75%) concentration and the other two bins at a 2-of-16 (12.5%) background concentration. Within that fixed structure, which actor or bin is the incident group, trace ordering, and the specific non-target fault occupying each background slot are all randomized per window. The target fault signature under investigation is rotated across all five fault classes across the validation and confirmatory windows, so no single result depends on one fault class happening to be unusually easy to classify.

2.6 Utility Metrics

Task output (what a solver returns) is defined separately from **utility scoring** (how that output is compared against ground truth), and ground truth is consulted only by the scorer, never by any solver.

For T1 and T3: binary correctness (predicted service/fault class equals ground truth). For T2: edge-set F1 against the true call graph. For T4 (latency-relevant faults only): binary correctness of the predicted root-cause span.

For T5 and T6: **top-1 identification accuracy** — whether the solver correctly attributes the incident cohort (T5) or bin (T6), judged by underlying membership rather than by comparing surface tokens directly — and **affected-set F1** — how completely the solver recovers the specific affected traces. A window is scored as **indeterminate** when no representation of the task's required signal survives the applied treatment at all (Section 2.5) — this is reported as its own outcome, never silently folded into a "wrong answer" by fabricating a spurious grouping.

For M4 affected-set F1 specifically, two figures are reported and kept conceptually distinct: **operational** F1, scored against the *full original* affected set — so a truly-affected trace that was dropped by sampling counts as a false

negative — and **conditional** F1, scored only among the traces that survived sampling. Operational F1 is the primary figure throughout this paper; conditional F1 is reported as a secondary, mechanism-illustrating figure. Top-1 identification is reported as a single outcome under M4, not split into operational and conditional variants. This operational/conditional distinction is what allows M4's effect to be interpreted later as loss of evidence completeness rather than corruption of retained content.

2.7 Exposure Metrics

Five metrics, defined independently of the utility metrics above and never influencing task scoring:

- **E1 — Plaintext retention**: the fraction of a trace's Class A/B/C values that survive a treatment unchanged, relative to the untreated (M0) trace.
- **E2 — Direct-identifier exposure**: whether any Class A value survives a treatment in a form matching its original plaintext value.
- **E3 — Cross-trace actor linkability**: whether two independently treated traces belonging to the same underlying actor can be linked under a given treatment. This metric exists specifically because plaintext disappearing does not, by itself, mean an actor's activity is unlinkable: **M2's deterministic pseudonymization of Class A is designed to preserve this property intentionally**, and we do not treat that as a privacy failure — it is a distinct, deliberate design point, not an oversight. M5, which suppresses Class A outright rather than pseudonymizing it, does not preserve actor linkability; its pseudonymization of Class B affects trace/span structural identifiers only, which is a separate property from actor-level linkage.
- **E4 — Sensitive-content retention**: the narrower, Class-C-only version of E1, isolating payload/business-context exposure from identifier exposure.
- **E5 — Corpus-level retained exposure**: relevant specifically to M4, this captures that a sampling treatment can reduce total exposure across a stored corpus (because some records are simply absent) even when the content of every surviving record remains fully exposed — a distinct axis from E1, which measures per-record content exposure only.

2.8 Instrument Validation and Freeze Discipline

Every diagnostic task (Section 2.5) was validated as a measurement instrument on unminimized (M0-only) data, against a pre-specified numeric acceptance threshold, before any minimization treatment was ever evaluated against it. T1 and T3 were validated against a 90% per-fault-class accuracy gate across 250 M0-only trials; T2 was required to achieve exact edge-set equality (F1 = 1.00) on every trial, since complete topology is directly readable from unminimized data and anything less indicates an instrument defect rather than acceptable task difficulty; T4 was validated against the same 90% gate restricted to its three applicable fault classes; T5 and T6, given their fully controlled, unambiguous batch construction (Section 2.5), were held to an exact top-1 = 1.00 and affected-set-F1 = 1.00 gate across 50 M0-only validation windows each, rotated across all five fault classes.

Validation used disjoint, pre-specified development and holdout seed ranges per task, and — where a task's first candidate implementation did not clear its holdout gate on the first attempt — a documented procedure of diagnosing on development data only, then confirming on a separate, never-previously-queried secondary holdout. Once validated, each task's exact algorithm, dependencies, and validation results were recorded as a structured instrument fingerprint (a canonical hash of the relevant solver, scorer, and ground-truth-derivation code), protected against silent modification: any change to a frozen instrument requires an explicit, reasoned amendment rather than a direct edit. **No frozen instrument was tuned to improve treatment-stage performance.** The only post-treatment instrument change was Amendment v1.4, a documented correction of a scoring defect exposed by an internal inconsistency in an exploratory pilot (Section 2.9): the underlying treated data remained valid throughout and were re-scored after the amended scorer was independently revalidated on M0-only data, without any change to which windows were generated or how treatments were applied to them.

Four such amendments were made over the course of this study, each triggered by a specific, reproducible finding rather than a disappointing or merely surprising result:

- **Removal of timestamp coarsening from the generalization treatment.** An early implementation draft had the generalization treatment coarsening span timestamps; since timing is explicitly outside this study's threat model (Section 2.3), this was corrected before any pilot data existed.
- **Formal classification of a retry-count signal as status/operational metadata (Class F).** Added during task-solver development to give the failure-class discrimination task a principled operational signal for distinguishing two fault classes whose duration distributions genuinely overlap, rather than relying on an arbitrary duration threshold.
- **Correction of the pseudonymization key-derivation implementation.** An early implementation derived the deterministic pseudonymization key per-trial rather than once per experiment, which defeated the intended cross-trace-linkability property (E3) the M2 treatment was designed to have. This was caught via an E3 analysis on exploratory pilot data showing near-zero linkage where the design called for consistent linkage, fixed to a genuine single experiment-level key, and the affected pilot data was invalidated and regenerated (Section 2.9).
- **Correction of T5's cohort-identification scoring to be representation-invariant.** An early scoring implementation compared a solver's selected cohort against ground truth by direct string comparison, which is well-defined under treatments that leave identity in plaintext but not under M2's deterministic pseudonymization, where the visible token is not the underlying identity value. This was caught via a pilot result showing perfect affected-set recovery alongside a reported top-1 failure under M2 — a combination only possible if the scoring itself, not the solver, was wrong — and corrected to compare underlying cohort membership rather than surface token identity.

Each amendment is documented with its trigger, root cause, and resolution in the project's experiment specification; we summarize rather than reproduce that record here.

2.9 Pilot and Confirmatory Protocol

Pilot v1 was invalidated and excluded from all subsequent empirical analyses. The per-trial M2 key-derivation defect meant its pseudonymization condition did not implement the frozen treatment design. The artifact is retained and clearly labeled for provenance rather than deleted.

Pilot v2, generated after that correction, evaluated Tasks 1–4 across all six treatments. Task performance under the field-level treatments (M1, M2, M3, M5) and conditional performance under M4 (scored on retained traces only) reached ceiling with no observed variance; M4's *operational* utility, in contrast, tracked trace retention rather than remaining at ceiling, consistent with sampling removing evidence rather than corrupting it. This motivated the design of Tasks 5 and 6 specifically to depend on the telemetry classes the field-level treatments actually target — full characterization of this result is deferred to Results.

Pilot v3 added two independent window arms — for the newly validated and frozen Tasks 5 and 6 — evaluated across all six treatments. Tasks 1–4's valid Pilot v2 arm was retained rather than regenerated: the additive simulator changes needed to support T5/T6's batch construction were verified to reproduce Pilot v2's original output exactly, byte-for-byte, before Pilot v3 began, so no scientific purpose would have been served by rerunning it. Pilot v3's T5/T6 data provided the variance and effect-size estimates used to size the confirmatory experiment.

Stage 2 is the confirmatory experiment, pre-specified and frozen in the project's version-controlled experiment specification before any Stage-2 data was generated: 500 independent windows per task (100 per fault class, rotated across all five), drawn from seed ranges never used in any prior validation, development, or pilot stage — T5 from 10000–10499, T6 from 11000–11499. Pilot findings informed Stage-2 sample size and endpoint selection; no aspect of the confirmatory design, including its primary endpoint, was chosen or adjusted after Stage-2 data existed.

2.10 Statistical Analysis

Within each task, treatment effects are computed as paired differences for every base window, since every treatment is derived from the same underlying base trace(s) for that task. T5 and T6 use independently generated windows and are never treated as paired with each other; any comparison between them uses independent-sample methods, consistent with the window being the unit of inference for both tasks (Section 2.1).

The single pre-specified primary confirmatory endpoint is defined as follows. For task *t* in {T5, T6}, let

$\Delta\_t = F1(M4, t) - F1(M0, t)$

be the per-window operational affected-set-F1 change under M4 relative to the M0 control, paired within task. The primary contrast is

$D = \Delta\_T5 - \Delta\_T6.$

Because Δ is defined as M4 minus M0, $D < 0$ indicates that the M4-induced operational-F1 reduction is larger for T5 than for T6; $D > 0$ indicates the reverse.

D is evaluated with a two-sided test at $\alpha = 0.05$. A 95% confidence interval on the observed D is obtained by independently resampling, with replacement, the window-level Δ_T5 values and the window-level Δ_T6 values (20,000 resamples, RNG seed 42), differencing the two resampled means on each iteration, and taking the 2.5th and 97.5th percentiles of the resulting distribution of 20,000 differences. The associated p-value is obtained by an independent-sample permutation test: the pooled Δ_T5 and Δ_T6 values are repeatedly shuffled and re-split into groups of the original T5 and T6 sizes (50,000 permutations, RNG seed 43), and the p-value is the proportion of permuted differences D_perm satisfying $|D\_perm| \geq |D\_obs|$, where D_obs is the observed value of D. Both procedures are simulation-based rather than a normal approximation, given the paired-within-task, non-normally-distributed nature of the underlying per-window Δ values. All remaining comparisons — the M4 top-1 task contrast, the M5 capability-split comparison, per-fault stratification, conditional-vs-operational F1, and exposure metrics — are reported as secondary analyses, not used to declare the confirmatory result.

Where a treatment produces the same value on every observed window (most clearly, M1's and M5's effect on T5, and M5's effect on T6), this is reported as **observed invariance across every confirmatory window and fault stratum tested**, not interpreted as proof of population certainty. A bootstrap interval computed on data with zero empirical variation is necessarily zero-width by construction; that reflects the absence of variation within the tested conditions, not a claim about conditions outside them.

3. Results

3.1 Confirmatory Primary Result

The pre-specified primary confirmatory endpoint is the difference between T5 and T6 in M4-induced operational affected-set-F1 change relative to M0 (Section 2.10): $\Delta\_t = F1(M4, t) - F1(M0, t)$, $D = \Delta\_T5 - \Delta\_T6$.

Across 500 independent confirmatory windows per task: $\Delta\_T5 = -0.3791$ (SD = 0.2320), $\Delta\_T6 = -0.3394$ (SD = 0.1321). The observed difference is $D = -0.0396$, with a 95% bootstrap confidence interval of $[-0.0632, -0.0161]$ (20,000 resamples) and a permutation-test p-value of 0.00110 (50,000 permutations), against a pre-specified $\alpha = 0.05$. Under the pre-specified $\alpha = 0.05$ decision rule, H0 was therefore rejected.

The confirmatory data provide evidence of a task-specific difference under the pre-specified analysis: the confidence interval excludes zero and the permutation p-value is below $\alpha = 0.05$. The observed cross-task difference was modest in magnitude ($D = -0.0396$) relative to the task-specific M4 operational affected-set-F1 losses (both exceeding 0.33). Section 3.3 examines the mechanism behind this difference directly.

3.2 Task-Specific Treatment Effects

Table 1. T5 and T6 outcomes across all six treatments, N = 500 windows per task.

Treatment	T5 top-1	T5 operational F1	T5 indet. rate	T6 top-1	T6 operational F1	T6 indet. rate
M0	1.000	1.000	0.000	1.000	1.000	0.000
M1	0.000	0.000	1.000	0.000	0.000	1.000
M2	1.000	1.000	0.000	1.000	1.000	0.000
M3	1.000	1.000	0.000	1.000	1.000	0.000
M4	0.942	0.621	0.000	1.000	0.661	0.000
M5	0.000	0.000	1.000	1.000	1.000	0.000

Note: where indeterminate rate = 1.000, the zero utility entries denote absence of a task-capable prediction because the required representation was unavailable (Section 2.6); they should not be interpreted as 500 ordinary misclassifications.

M1 rendered both tasks indeterminate in all 500 windows. M2 and M3 preserved top-1 and operational F1 at 1.000 for both tasks. M4's effect is examined in Section 3.3. M5 is the one treatment that separates the two tasks completely — full detail in Section 3.4.

3.3 M4 Operational vs. Conditional Utility

M4's effect on both tasks is substantially different when measured on retained evidence alone versus measured against the full original affected set.

- **T5**: operational F1 = 0.6209, conditional F1 = 0.9420.
- **T6**: operational F1 = 0.6606, conditional F1 = 1.0000, with zero wrong-category selections across all 500 windows.

The gap between operational and conditional F1 shows that missing evidence accounts for most of M4's operational reduction. For T5, sampling additionally altered visible cohort concentrations sufficiently to produce wrong-cohort selection in 29 of 500 windows; no corresponding wrong-category selections occurred for T6. T5's six-way discrimination (Section 2.5) has narrower concentration margins than T6's three-way discrimination, which is consistent with T5 being the task where this secondary mechanism was observed.

3.4 M5 Capability Split

Under M5 (suppress Class A, pseudonymize Class B, generalize Class C, retain D/E/F), T5 and T6 diverge completely and without exception.

For T5 under M5, top-1 accuracy and operational affected-set F1 were both 0.000, with all 500 windows indeterminate — M5 suppresses the Class A actor identity T5 requires, leaving no representation of the required signal (Section 2.5). For T6, top-1 accuracy and operational affected-set F1 were both 1.000 across all 500 windows — M5's generalization of Class C leaves T6's required resource-category signal fully intact.

This split holds without a single exception across every one of the five 100-window fault strata for both tasks (Table 2):

Table 2. M5 operational F1 by fault class, N = 100 windows per fault class per task.

Fault class	T5 M5 operational F1	T6 M5 operational F1
dependency_timeout	0.000 (n=100)	1.000 (n=100)

latency_regression	0.000 (n=100)	1.000 (n=100)
authorization_failure	0.000 (n=100)	1.000 (n=100)
retry_amplification	0.000 (n=100)	1.000 (n=100)
partial_transaction	0.000 (n=100)	1.000 (n=100)

We report this precisely: **the M5 split was observed as invariant across all 500 confirmatory windows and all five fault strata under the frozen experimental design.** This is not framed as proof of population-level certainty — the invariance describes what was observed under the tested conditions, not a claim about conditions outside them.

3.5 Exposure Results (E1–E5)

Table 3. E1 (plaintext retention), E2 (direct-identifier exposure rate), and E4 (Class-C-only retention) by treatment, measured on Stage-2 data.

Treatment	E1 (plaintext retention)	E2 (direct-ID exposure)	E4 (Class-C retention)
M1	0.193	0.000	0.000
M2	0.420	0.000	1.000
M3	0.580	1.000	0.000
M4	1.000	1.000	1.000
M5	0.000	0.000	0.000

E3 (cross-trace actor linkability) and E5 (corpus-level retained exposure) are Pilot-v2 treatment-level measurements, not Stage-2 task-specific estimates — these characterize what a treatment exposes about trace content independent of which diagnostic task is applied (Table 4).

Table 4. E3 (cross-trace actor linkability) and E5 (corpus-level retained exposure) by treatment, Pilot v2.

Treatment	E3 (Pilot v2)	E5 (Pilot v2, M4 only)
M1	0.000	—
M2	1.000	—
M3	1.000	—
M4	1.000 (among retained traces)	0.500
M5	0.000	—

Two results in Table 4 are worth stating precisely rather than collapsing into a single "linkable/not linkable" judgment. First, M2 and M3 both yielded E3 = 1.000 through different mechanisms: M2 preserved actor equality through deterministic pseudonymization, whereas M3 left Class-A identifiers unchanged. Identical linkability outcomes, different underlying privacy properties. Second, under M4, E3 (1.000, among the traces that survive sampling) and E5 (0.500, observed in Pilot v2 across the full original corpus) report genuinely different quantities: content exposure conditional on survival is total, while Pilot v2's E5 was 0.500 across the corpus as a whole. Neither number alone characterizes M4's exposure profile.

3.6 T1–T4 Preservation and Pilot Context

Tasks 1–4 rely primarily on structural, status, timing, and graph-correlation information rather than the Class-A actor identity or Class-C payload/business-context fields targeted by suppression and generalization. Where Class-B identifiers (trace/span/parent-span structure) are pseudonymized under M2 or the pseudonymization component of M5, parent-child equality is preserved by construction (Section 2.4), so topology-dependent tasks are unaffected even though they do read Class-B fields.

Pilot v2 produced utility of 1.000 with zero observed variance in all 108 applicable task × treatment × fault cells. For M4, the ceiling calculation included retained traces only; dropped traces had no utility value and were excluded from that calculation. Thus, the Pilot-v2 ceiling should be interpreted as preservation of diagnostic capability when the required trace evidence remained available, not as absence of an operational cost from trace sampling.

This ceiling is reported as a positive, informative result: four distinct diagnostic tasks, spanning failing-service identification through latency localization, remain fully functional under suppression, pseudonymization, generalization, and the composite treatment. It is this finding — that an entire class of diagnostic tasks was unaffected by every field-level treatment tested — that motivated designing Tasks 5 and 6 specifically to depend on the telemetry classes those treatments actually target (Section 2.5).

3.7 Results Summary

Across the evaluated diagnostic tasks, the minimization treatments produced different utility patterns rather than a uniform response. In Pilot v2, T1–T4 retained ceiling diagnostic performance under the field-level treatments and on M4-retained traces. In the Stage-2 confirmatory experiment, M2 and M3 preserved both T5 and T6 at top-1 accuracy and operational affected-set F1 of 1.000, whereas M1 rendered both tasks indeterminate in every window. M5 produced the strongest observed task split: T5 was indeterminate across all 500 windows, while T6 retained top-1 accuracy and operational affected-set F1 of 1.000 across all 500 windows and each of the five fault strata.

Under M4, operational affected-set F1 decreased for both window-level tasks. The pre-specified cross-task contrast was $D = -0.0396$ (95% bootstrap CI $[-0.0632, -0.0161]$, permutation $p = 0.00110$), indicating a larger operational reduction for T5 under the defined contrast. Conditional analysis showed that evidence loss accounted for most of the M4 reduction, while T5 additionally produced wrong-cohort selections in 29 of 500 windows; no corresponding wrong-category selections occurred for T6. Separately, the exposure measurements distinguished plaintext retention, cross-trace actor linkability, and corpus-level retained exposure across the minimization mechanisms.

4. Discussion

4.1 Task-Dependent Diagnostic Sufficiency

We can now answer the research question directly: the same minimized trace representation can remain sufficient for one diagnostic task while becoming insufficient for another. M5 is the cleanest illustration of this — a single treatment that renders T5 indeterminate while fully preserving T6, invariantly across every confirmatory window and fault stratum tested (Section 3.4). M4 provides complementary confirmatory evidence through a different mechanism: whole-trace sampling, which never alters the content of any surviving record, still produces a significant, task-specific difference in how much diagnostic completeness is lost (Section 3.1).

Taken together, these are two structurally different demonstrations of the same phenomenon — one from content removal, one from volume reduction — which is what makes the combined result more than a single anecdote about one treatment and one task pair.

4.2 Different Minimization Mechanisms Fail Differently

The four content-oriented and volume-oriented mechanisms evaluated here degrade diagnostic capability through distinct failure modes, not a shared "more privacy, less utility" gradient:

- **Suppression** removes a signal entirely. When a task depends on the removed class, the result is not degraded performance but indeterminacy — the task has no basis for a prediction at all (Section 2.6), which is a qualitatively different failure than getting an answer wrong.
- **Pseudonymization** can preserve equality-dependent diagnostic capability while removing plaintext. In this study, M2 preserved the stable actor-linkage relation required by T5 while removing the original Class-A plaintext identifiers.

- **Generalization** preserves category-level reasoning while removing exact values. T6, which needs to know *which bin* a value falls into rather than its exact value, survives generalization completely. We did not evaluate a task requiring an exact Class-C value, so we make no claim about whether such a task would survive generalization — only that the one task tested did not need that precision.
- **Sampling** does two distinct things at once: it removes evidence outright (the dominant effect for both T5 and T6), and it can alter the visible composition of a group closely enough to change which group a concentration-based decision favors (the secondary mechanism observed for T5 only, Section 3.3).

Treating "minimization" as one axis obscures this — the mechanism matters as much as its magnitude.

4.3 Interpreting the M4 Confirmatory Effect

We want to be precise about what the M4 result does and does not establish. $D = -0.0396$ is statistically well-supported (95% CI $[-0.0632, -0.0161]$, $p = 0.00110$) but modest in absolute size relative to either task's own loss under M4 (both exceeding 0.33). The contribution is not that sampling hurts one task dramatically more than the other in some large, dramatic sense — it does not. The contribution is narrower and, we think, more interesting: **even identical, content-blind, whole-trace sampling at the same rate does not impose identical diagnostic loss across tasks.** Under this frozen design and a sampling rate of $p = 0.50$, the hypothesis that M4 induces equal operational affected-set-F1 loss for T5 and T6 was rejected at the pre-specified $\alpha = 0.05$ (Section 3.1). We do not claim this generalizes to other sampling rates, which this study did not test (Section 4.7).

The 29-of-500 T5 wrong-cohort selections (Section 3.3) are the concrete mechanism behind part of this difference: T5's six-way discrimination has narrower concentration margins than T6's three-way discrimination, so the same missing-evidence process is more likely to occasionally flip which group looks most concentrated. This is a structural property of task granularity interacting with sampling noise — and it is itself a task-dependent effect, consistent with this paper's central claim rather than a separate phenomenon.

4.4 Interpreting the M5 Capability Split

T5 requires persistent actor-identity correlation; T6 requires resource-category information. M5 suppresses the former and generalizes the latter, by design (Section 2.4) — so the split is, in an important sense, exactly what the treatment's own definition predicts. We are explicit about this rather than presenting it as a surprising universal law: the divergence follows directly from how M5 was defined relative to what T5 and T6 each need, not from an emergent property discovered independent of the experimental design.

The predicted M5 split was observed without exception across all 500 confirmatory windows and five fault strata, while remaining a property of the tested design. T5's identity requirement, T6's category requirement, and M5's treatment semantics (Section 2.4, Section 2.5) were fixed before T5 and T6 were evaluated under minimization treatments and before any Stage-2 data existed. The direction of the split was therefore expected; the empirical result is that it persisted without an observed exception across the full confirmatory design.

4.5 Privacy–Utility Implications

The exposure results (Section 3.5) show that "privacy" itself resists collapsing into one scalar quantity, in a way that matters for how these results should be read. M2 and M3 produce identical cross-trace-linkability outcomes (E3 = 1.000) through different mechanisms — deliberate, designed pseudonymization for M2, versus simply never touching the field for M3 — meaning identical exposure numbers on one metric can reflect very different privacy postures. Symmetrically, M4 shows that reducing corpus-level exposure (E5 = 0.500, Pilot v2) and reducing per-record content exposure are different axes entirely: every trace that survives sampling is as fully exposed as it was in M0, even though the stored corpus as a whole contains less.

The practical consequence is that neither "does this treatment reduce average exposure" nor "does this treatment preserve average utility" is a well-formed question without specifying which exposure dimension and which diagnostic task are meant.

4.6 Relation to Prior Work

Our results are consistent with, and extend, several strands of prior work discussed in Section 1.3, without claiming those strands ignored utility or privacy.

Proteus [3] and Zhou et al. [4] each recognize, in their own domains, that minimization can remove information a downstream purpose needs; our contribution is not that recognition itself, but a systematic, task-differentiated measurement of when and how much that removal matters, evaluated against established mechanisms rather than a novel one. PrvTel [6] and Mohammady et al. [5] establish that privacy-utility tradeoffs for telemetry- and network-trace data can be rigorously quantified; our results extend that discipline to application-level distributed traces and, specifically, to whether the same minimized representation preserves different diagnostic tasks differently — an interaction their utility formulations were not organized to evaluate. AutoScope's [7] finding that trace reduction can preserve downstream RCA usefulness is a close methodological neighbor to our M4 results, though motivated by storage cost rather than privacy and without an exposure axis; RCAEval [8] reinforces that diagnostic utility should be measured as explicit, scored task performance rather than assumed from telemetry availability — the same discipline this paper applies to its own task solvers. Contrast's [9] finding that diagnostically useful differences between distributed traces can manifest across structural, temporal, critical-path, and semantic dimensions is conceptually consistent with, and independent evidence for, treating trace utility as multidimensional, though privacy minimization is outside its problem entirely.

Say Something Else [10] studies a different interaction. Its variance decomposition attributes 98.4% of variation in its IS-AD measure to scenario-level differences, compared with 1.0% to strategy and 0.2% to model identity, while still finding statistically significant strategy effects. In our setting, the interaction is different: we hold the trace representation and minimization treatment fixed and ask whether that representation can remain sufficient for one diagnostic task while becoming insufficient for another. The two findings are therefore complementary rather than directly analogous.

4.7 Limitations and External Validity

We state these directly rather than minimizing them.

- **Synthetic, in-process simulator**, not a deployed distributed system (Section 2.2). Results characterize the experimental design we built, not a claim about production telemetry pipelines directly.
- **Five services, a controlled topology**, and **deterministic synthetic faults** — a small, fixed universe of failure conditions, not a sample from real production incident distributions.
- **Six diagnostic tasks and five minimization treatments** — a fixed, finite set, chosen to span structural, identity, and content dependencies, but not an exhaustive catalog of either diagnostic tasks or minimization techniques.
- **M4 was evaluated only at a single sampling rate, p = 0.50.** We do not know how the cross-task operational-F1 gap observed here would behave at other sampling rates — whether it narrows, widens, or changes shape as the retention rate varies is outside what this study tested.
- **Timing (Class E) was deliberately placed outside this study's threat model** (Section 2.3); we make no claim about timing-based side channels or timing-sensitive privacy risks.
- **No exhaustive search over telemetry subsets was performed for any task.** We show that specific, established mechanisms are sufficient or insufficient for specific tasks; we do not claim to have found, or searched for, the provably minimal representation for any task.

- **T5 and T6's batch construction uses controlled, non-probabilistic concentration margins** (Section 2.5), chosen so that task difficulty would not be determined by sampling luck. This is a deliberate design choice, not a claim that real-world incident cohorts or resource-category distributions exhibit these exact concentration ratios.
- **The M5 divergence is, as discussed in Section 4.4, partly structural by construction** — it follows from how M5, T5, and T6 were each defined, not from an independently emergent discovery.
- **Although the T4 self-time implementation handles overlapping child intervals, the synthetic topology does not contain a span with multiple independent child branches; consequently, the multi-child merge path was not exercised empirically.** Any future topology introducing such a span would require revalidating that branch before trusting T4's results under it.
- **Diagnostic utility is measured through frozen algorithmic tasks rather than human-operator performance.** The results therefore characterize whether specified diagnostic capabilities remain recoverable from a minimized representation, not how quickly or accurately human practitioners would diagnose incidents using that representation.
- **This study does not measure production-scale latency, storage, or computational overhead** for any minimization treatment, nor does it evaluate these mechanisms within a real organizational incident-response workflow.

4.8 Practical Implications / Design Guidance

The actionable implication is not a specific universal rule about which fields to keep. It is a process claim: **telemetry minimization policy should begin from the diagnostic capabilities that must be preserved, then determine which telemetry representations those specific capabilities actually require** — rather than starting from a generic notion of "sufficient telemetry" and applying one minimization policy uniformly across every downstream use. Our results show concretely that treatments serving one goal do not automatically serve another: M2 reduces plaintext exposure while deliberately preserving the cross-trace linkage T5 depends on, whereas M3 leaves Class-A identifiers untouched while removing the exact Class-C values that are not what T6 actually needs. A policy chosen to reduce one exposure dimension does not thereby guarantee, or preclude, preservation of any particular diagnostic capability — the two have to be reasoned about separately. We do not prescribe which fields any particular system should retain — that depends on which diagnostic tasks that system's operators actually need, which is an organization-specific question this study does not attempt to answer.

4.9 Discussion Summary

The results support a task-dependent view of diagnostic sufficiency: a telemetry representation that is sufficient for one diagnostic capability need not be sufficient for another, even when both operate over traces produced by the same system and undergo the same minimization treatment. This does not establish a globally minimal trace schema; rather, it shows that utility preservation cannot be treated as a task-independent property of a minimized trace representation. The mechanisms behind this — content removal producing indeterminacy, pseudonymization preserving stable equality and linkage relationships, generalization preserving task-relevant categorical information, and even content-blind sampling producing task-differentiated evidence loss — together indicate that minimization policy and diagnostic-capability requirements need to be reasoned about jointly, not treated as separable design decisions.

5. Conclusions

This paper asked whether privacy-driven minimization of distributed-trace data affects task-specific diagnostic utility uniformly, or whether different diagnostic tasks require different minimum telemetry representations. Across six independently defined and independently validated diagnostic tasks and five minimization treatments, the results support a task-dependent view of diagnostic sufficiency: a representation sufficient for one diagnostic capability need not be sufficient for another.

Two results carry this finding. Under a composite treatment that suppresses actor identity while generalizing payload and business context, cross-request incident-cohort attribution was rendered indeterminate on every one of 500 confirmatory windows and all five fault strata, while context-associated failure diagnosis remained fully intact on the same 500 windows and strata — a single composite treatment, applied to the same underlying telemetry, with different consequences for the two diagnostic capabilities. Through a different minimization mechanism, whole-trace sampling — which never alters the content of a surviving trace — produced a statistically significant task-specific difference in operational affected-set-F1 loss between T5 and T6 ($D = -0.0396$, 95% CI $[-0.0632, -0.0161]$, $p = 0.00110$). This shows that task-dependent diagnostic loss can arise even when minimization changes telemetry availability rather than the content of retained traces.

The practical implication is that telemetry minimization cannot be designed as a single, task-agnostic policy: what representation is "sufficient" depends on which diagnostic question is being asked of the data, and a policy tuned to protect or preserve one diagnostic capability may do nothing for another. We do not claim to have identified a globally minimal trace schema, and this study does not attempt to enumerate one — our results establish that utility preservation is a task-dependent property of a minimized representation, not that any particular minimal representation exists or has been found. Minimization policy and diagnostic-capability requirements need to be reasoned about jointly, on a capability-by-capability basis, rather than as a single design decision applied uniformly across every downstream use.

6. Funding

This research received no external funding.

7. Author Contributions

Lokeswari Vejju conceived and designed the study, developed the experimental methodology and synthetic simulator, performed the experiments and statistical analysis, interpreted the results, conducted the literature review, and prepared and revised the manuscript.

8. Conflict of Interest

The author declares no conflict of interest.

9. Data Availability Statement

The synthetic data and supporting experimental artifacts generated for this study are available from the author upon reasonable request.

Biographical Sketch

Lokeswari Vejju is a Software Development Engineer II at Amazon Web Services (AWS) with more than six years of experience in secure cloud platforms, cryptographic infrastructure, distributed systems, and observability for regulated environments. Her professional experience spans AWS Cryptography, Goldman Sachs Core Engineering, and Citi, with technical work involving hardware-backed cryptographic systems, distributed tracing and observability, performance engineering, and secure financial infrastructure. She holds an M.S. in Computer Science from Arizona State University and a B.Tech. in Information Technology from the National Institute of Technology Allahabad. Her research interests include secure and trustworthy distributed systems, privacy-aware observability, cryptographic infrastructure, and diagnostic reliability in complex software systems.

REFERENCES

1. OpenTelemetry Authors, "Tracing API," OpenTelemetry Specification. Available at: https://opentelemetry.io/docs/specs/otel/trace/api/ (Accessed on 10 August 2026).
2. OpenTelemetry Authors, "Handling sensitive data," OpenTelemetry Documentation. Available at: https://opentelemetry.io/docs/security/handling-sensitive-data/ (Accessed on 10 August 2026).

3. S. Goutam, H. Kippen, M. Grace, A. Rahmati. "Proteus: A Practical Framework for Privacy-Preserving Device Logs", Proceedings on Privacy Enhancing Technologies, 2026(4), pp. 891-905, 2026. DOI: 10.56553/popets-2026-0150. Available at: https://doi.org/10.56553/popets-2026-0150
4. L. Zhou, J. Zou, L. Yu, H. Min. "Privacy-Preserving Redaction of Diagnosis Data through Source Code Analysis". In Proceedings of the 35th International Conference on Scientific and Statistical Database Management (SSDBM), 2023. DOI: 10.1145/3603719.3603734. Available at: https://doi.org/10.1145/3603719.3603734
5. M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, M. Debbabi. "Preserving Both Privacy and Utility in Network Trace Anonymization". In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 459-474, 2018. DOI: 10.1145/3243734.3243809. Available at: https://doi.org/10.1145/3243734.3243809
6. Y. Zhou, F. Zhao, E. Wang, A. K. Coskun, D. Agrawal, A. El Abbadi, Z. Liu. "PrvTel: Lightweight Models for Private and Accurate Telemetry Data Retention". In Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI '26), pp. 1827-1843, 2026. Available at: https://www.usenix.org/conference/nsdi26/presentation/zhou-yajie
7. Y. Wu, G. Yu, Z. Jiang, Y. Li, M. R. Lyu. "Trace Sampling 2.0: Code Knowledge Enhanced Span-level Sampling for Distributed Tracing". arXiv:2509.13852, 2025. Available at: https://arxiv.org/abs/2509.13852
8. L. Pham, H. Zhang, H. Ha, F. Salim, X. Zhang. "RCAEval: A Benchmark for Root Cause Analysis of Microservice Systems with Telemetry Data". In Companion Proceedings of the ACM Web Conference 2025 (WWW Companion), pp. 777-780, 2025. DOI: 10.1145/3701716.3715290. Available at: https://doi.org/10.1145/3701716.3715290
9. V. Anand, R. Fonseca, J. Mace, A. Kaufmann. "Enabling Multi-Dimensional Distributed Trace Comparison with Contrast". arXiv:2607.19102, 2026. Available at: https://arxiv.org/abs/2607.19102