Article

Not peer-reviewed version

RootTrace: Root Cause Localization for Settlement Discrepancies in Multi-Party Electricity Trading via Trusted Path Traceability

Pingyan Mo *, Kai Li , Xihong Liang , Jiajun Liu , Xin Hu , Jinwen Xi *

Posted Date: 22 April 2026

doi: 10.20944/preprints202604.1623.v1

Keywords: electricity market settlement; reconciliation; settlement discrepancy; trusted path traceability; root cause analysis

 doi:10.20944/preprints202604.1623.v1

Article

RootTrace: Root Cause Localization for Settlement Discrepancies in Multi-Party Electricity Trading via Trusted Path Traceability

Pingyan Mo [1,*], Kai Li [1], Xihong Liang [1], Jiajun Liu [2], Xin Hu [2] and Jinwen Xi [3,*]

[1] Digital Intelligence Operation Center of Guangdong Power Grid Co., Ltd., China
[2] Guangdong Power Trading Center Co., Ltd., China
[3] Beijing University of Posts and Telecommunications, Beijing, China
* Correspondence: mopingyan@xxzx.gd.csg.cn (P.M.); jwxi@bupt.cn (J.X.)

Abstract

Settlement discrepancies in multi-party electricity trading are difficult to localize because final outcomes are produced by multi-stage pipelines that combine heterogeneous data, rule versions, parameters, and execution contexts across organizational boundaries. In such settings, numerical reconciliation is not enough: investigators must identify where divergence entered the pipeline and support that judgment with evidence that can be checked independently. We formulate discrepancy localization as an auditable inference problem and introduce *RootTrace*, a trusted path-traceability method for this setting. Settlement processing is represented as an evidence graph over versioned artifacts and explicit events; RootTrace uses backward tracing and version-difference tracing to derive a suspect set of stages and/or artifact versions. The same pipeline exports a verifiable evidence bundle that preserves the trace used in localization. To support accountability under an explicit threat model, RootTrace includes trusted recording and verification procedures for tamper-evident capture and minimal-disclosure bundle export. On a semi-synthetic benchmark with ten independent replications, RootTrace achieves mean Top-1/Top-5/MRR of 0.47/0.83/0.60, compared with Top-1 0.17 and 0.04 for representative rule-based and stage-order baselines; exported bundles verify cleanly with full detection of the configured tamper classes (mean verification latency below 5 ms on our grid); and an eight-hour unattended stress run completes over 1.3×10^6 iterations without runtime failure.

Keywords: electricity market settlement; reconciliation; settlement discrepancy; trusted path traceability; root cause analysis

1. Introduction

Settlement discrepancies in multi-party electricity trading can trigger costly reconciliation and dispute-handling processes because final payable and receivable amounts are produced by a multi-stage pipeline rather than by a single isolated computation. Cross-cutting energy-system research highlights demand-side mitigation pathways [1], renewable-led global transformation and its technical bottlenecks [2], and long-horizon European electricity-grid expansion planning [3]. Empirical market studies, from California's restructuring experience [4], through technology-transition mechanisms in Australia's National Electricity Market [5], to welfare-oriented analysis of demand-side participation in PJM [6], similarly show that wholesale outcomes arise from layered rules, charges, and governance processes tied to metering, pricing, and settlement-related data paths rather than from any single isolated computation. Consequently, when a discrepancy is observed, investigators must answer not only how much two outcomes differ, but also where in the processing path that difference was introduced.

The core problem is therefore *discrepancy localization*: given two comparable settlement outcomes that differ, determine which stage, artifact version, or execution difference most plausibly introduced

 doi:10.20944/preprints202604.1623.v1

that divergence, and do so in a way that can be justified to multiple parties. This problem is especially challenging in multi-party settings. Settlement processing often crosses organizational and system boundaries, different parties may see only part of the evidence, and outputs are sensitive to both data quality and version choice. A discrepancy may be caused by missing or duplicated records, time-alignment mismatches, inconsistent aggregation, rule drift, parameter updates, or execution drift between otherwise comparable runs. The problem is therefore structural: the same output symptom may be compatible with several upstream causes unless the full processing path is reconstructed carefully.

Existing research addresses important parts of this challenge, but not the whole of it. Provenance and lineage support dependency tracing across entities, activities, and versions [7–10]; secure logging supports tamper-evident evidence capture and auditability [11–13]; and root-cause analysis supports candidate reduction over logs and dependency structures [14]. However, these strands do not by themselves provide a settlement-oriented workflow that starts from an observed discrepancy, localizes the likely stage or version divergence, and exports a verifiable evidence package suitable for dispute handling under cross-party access constraints.

Challenges. Conventional reconciliation workflows and ordinary system logs remain useful for detecting numerical differences, but they are often weaker at supporting *verifiable* accountability. In dispute settings, it may be insufficient for one party to assert that a particular dataset, rule version, or parameter set was used. The investigation may also need to withstand questions about whether records were altered, whether a different version was actually applied, whether the reported trace is complete within the stated system boundary, and how much sensitive information must be disclosed to justify the conclusion. These requirements jointly define the main challenge considered here: localization must be useful, verifiable, and privacy-aware at the same time.

The central question is whether an observed settlement discrepancy can be localized to a likely divergence stage or version in a form that withstands third-party checking. We address this through *RootTrace*, the proposed evidence-driven localization method based on trusted path traceability. In this framing, localization is not only a debugging aid; it is an auditable inference task whose output must include both a suspect set and the trace that supports it.

RootTrace combines four elements: an explicit problem and threat formulation, an evidence model over versioned artifacts and events, a trusted recording and verification layer, and a localization pipeline that returns ranked candidates together with an exportable evidence bundle. The point of that integration is to narrow the gap between "finding a plausible cause" and "defending that conclusion in a multi-party audit setting."

Contributions. The paper makes three main contributions:

- We formalize discrepancy localization for multi-party electricity settlement through an explicit query model, output specification, evaluation metrics, and threat model, thereby defining the problem scope and the evidential requirements that later design choices must satisfy.
- We develop *RootTrace*, a localization method built around a Settlement Discrepancy Propagation Chain (SDPC) model, an evidence-chain model, a minimal-sufficient-evidence criterion, a trusted recording and verification mechanism, and an evidence-driven workflow that outputs both a suspect set and a verifiable evidence bundle under minimal disclosure.
- We instantiate *RootTrace* in a reference implementation and evaluate it on the semi-synthetic benchmark and protocol of Section 6 (ten independent replications). Under frozen hyperparameters, RootTrace reaches mean Top-1/Top-5/MRR of 0.47/0.83/0.60, outperforming incremental baselines such as rule-based checking and stage-order backtracking (e.g., B1 Top-1 0.17; B2 Top-1 0.04). Bundle verification accepts all clean exports and flags all configured tamper classes in our grid, with mean per-bundle verification latency below 5 ms and serialized evidence on the order of 35 kB per bundle; removing SDPC propagation collapses Top-1 to 0.00 in ablation, while disabling trust preserves localization but reduces tamper detection from 1.00 to 0.86. An eight-hour unattended stress run completes over 1.3×10^6 iterations with zero runtime failures.

 doi:10.20944/preprints202604.1623.v1

The rest of the paper is organized as follows. Section 2 reviews the most relevant background and related work. Section 3 defines the query model, scope, and threat assumptions. Section 4 presents the localization method, and Section 5 gives the evidence recording and verification mechanism. Section 6 reports the experimental setup and results. Section 7 discusses deployment implications and limitations, and Section 8 concludes.

2. Background and Related Work

2.1. Settlement Discrepancy Investigation Background

Electricity-market settlement converts measured operations and contractual rules into payable/ receivable results for a defined time window. Related wholesale-market evidence, California's restructuring episode [4], Australian National Electricity Market (NEM) transition and market mechanisms [5], and PJM Interconnection (PJM) demand-response welfare interactions with wholesale prices [6], is consistent with treating payable outcomes as the result of staged rules, charges, and governance processes (including metering, pricing, and publication/reconciliation steps) rather than as a single-shot calculation. This operational structure implies that a published settlement outcome depends on chained transformations rather than on a single isolated computation.

Accordingly, the discrepancy considered here is not only a numeric mismatch. It is an observed divergence between two comparable outcomes under the same business scope, time window, and interpretation rule, where the divergence must be explained along the upstream processing path. In practice, this path can involve data preparation, rule/parameter application, execution scheduling, and publication or reconciliation actions. These observations motivate a version-aware, stage-aware model as the basis of investigation.

2.2. Related Work and Limitations

RootTrace draws on four literature strands. Each contributes something important, but none resolves the target problem on its own.

Provenance and lineage for multi-stage processing. The W3C PROV model provides a practical representation centered on entities, activities, and agents [15]. Recent database and platform lineage systems extend that view to large-scale pipelines and complex analytics, including dynamic coarse-grained provenance extraction [7], augmented lineage for user-defined functions (UDF) [8], and unified industrial lineage services [9]. Forensic provenance reconstruction for operational intrusions further illustrates end-to-end trace demands [10]. This strand directly supports backward trace reconstruction but does not by itself guarantee tamper evidence or third-party verifiability.

Root cause localization from logs and dependency structures. Root cause analysis (RCA) literature reduces large candidate spaces through execution comparison and structured dependencies; relational debugging is a representative example [14]. That line of work is useful for candidate narrowing and trace differencing, but its outputs are usually designed for internal diagnosis rather than cross-party evidence exchange.

Trustworthy recording and minimal-disclosure verification. In multi-party dispute settings, an explanation is useful only if the underlying records can be trusted. Recent system-audit architectures reiterate why high coverage, synchronous availability, and isolation against privileged producers matter for trustworthy logs [11,12], while certificate transparency and related append-only log designs encode publicly verifiable Merkle semantics for integrity-evidence exchange [13]. These mechanisms provide the integrity foundation required for settlement investigations.

At the same time, cross-organizational audits often cannot disclose all payload data. Privacy-oriented transparency systems indicate that commitment-based verification can coexist with selective disclosure when interfaces are designed explicitly [16]. For settlement discrepancy analysis, this means that graph-linking identifiers and integrity-relevant fields must remain verifiable, while sensitive payloads can be protected if their commitments and required metadata are retained.

 doi:10.20944/preprints202604.1623.v1

2.3. Research Gap, Differences, and Advantages

The gap is not the absence of relevant techniques; it is the absence of a workflow that joins them in a form appropriate for settlement disputes. Wholesale-market and integration studies, California restructuring [4], the Australian NEM [5], PJM demand response [6], and interconnection-driven integration in the Irish Single Electricity Market [17], document multi-stage rules, coupling, and governance pressures that motivate version-aware dispute analysis; provenance and lineage systems support dependency reconstruction [7–10,15]; secure logging research together with transparency-log designs provide tamper-evident recording and third-party verifiability [11–13]; and RCA methods reduce candidate spaces [14]. What remains underspecified is a settlement-oriented workflow that combines version-level localization, independently verifiable evidence export, and minimal-disclosure support for cross-party dispute handling.

Table 1 summarizes the complementary strengths and gaps across these strands. Existing wholesale-market and integration studies may explain outcome differences at the rules-and-governance level, and tracing or logging techniques may record parts of the processing path, but a settlement dispute workflow also needs to identify where versions diverged, preserve that explanation in a replayable evidence subgraph, and let an independent verifier check the integrity of the trace under limited disclosure. RootTrace is intended to fill that integrated role.

Table 1. Position of RootTrace relative to related strands.

Strand (refs)	Version	Verify	Main gap
Wholesale / integration studies [4–6,17]	Partial	No	Process and governance only; no formal localization workflow
Provenance / lineage [7–10,15]	Yes	No	Good for dependency tracing, but not dispute-grade tamper evidence
Secure logging [11–13]	No	Yes	Protects records, but does not localize settlement discrepancies
RCA / debugging [14]	Partial	No	Narrows candidates, but lacks auditable evidence export
Limited disclosure [16]	No	Yes	Supports commitment checks, but not settlement-path localization

Against that background, RootTrace offers three practical advantages for settlement disputes.

Difference 1: From generic traceability to discrepancy-oriented localization. Rather than building a general provenance catalog, RootTrace takes a discrepancy query as input and returns a suspect set linked to concrete divergence points (data, rule/parameter, execution version, or publication stage), reducing investigator search space.

Difference 2: From internal diagnostics to independently checkable evidence. The output is not only a localization result but also a bundle that carries records, digests, signatures, and ordering material, enabling third-party verification under the defined trust model.

Difference 3: From all-or-nothing visibility to policy-driven disclosure. RootTrace explicitly separates mandatory disclosed fields from commitment-only protected fields, so cross-party verification remains feasible without requiring full raw-data exposure.

Taken together, these differences give RootTrace a more closed investigative workflow: discrepancy observation, trace reconstruction, suspect localization, and evidence verification are handled in one pipeline. That matters in low-trust multi-party settings, where the gap between technical diagnosis and defensible justification is often the real bottleneck.

3. Problem Definition and Threat Model

3.1. System and Discrepancy Query Model

We model settlement as a multi-stage pipeline $s \in \mathcal{S}$ over versioned artifacts. For localization, we distinguish five artifact classes: data D, rules R, parameters P, jobs J, and outputs O. In the dependency graph, each node is typed into $\{D, R, P, J, O\}$ according to its settlement role (e.g., raw inputs, rule versions, parameters, intermediate or corrective processing, published outputs). A concrete artifact

 doi:10.20944/preprints202604.1623.v1

version is denoted $a@v$, where v is a stable content-addressed identifier derived from canonicalized content plus minimal scope metadata (e.g., window and type). For time-windowed data, different extraction/alignment results are treated as different versions even if business labels are similar. A discrepancy investigation starts from $q = (W, U, Y^{(ref)}, Y^{(tar)}, \Delta, \mathcal{C})$, where W is the settlement window, U is scope, $Y^{(ref)}$ and $Y^{(tar)}$ are comparable outcomes, Δ is the observed difference, and $\mathcal{C}$ is optional context (e.g., run IDs or publication timestamps). The model is formula-agnostic: it does not assume a specific market equation, only that comparable outputs and their producing traces can be referenced.

Query abstraction. Operationally, a query supplies the settlement window W; optional scope U (participants, products, or line items); identifiers that resolve to one reference output $Y^{(ref)}$ and one target output $Y^{(tar)}$ under the same logical scope; a structured description of the observed discrepancy when available; and optional context such as run identifiers or publication times. The scalar gap Δ in $q = (W, U, Y^{(ref)}, Y^{(tar)}, \Delta, \mathcal{C})$ denotes the measured output difference used for logging and evaluation.

Localization is anchored on the resolved pair $(Y^{(ref)}, Y^{(tar)})$. If only coarse run identifiers are initially available, a registry maps them to concrete output versions. A valid query therefore yields exactly one comparable reference–target pair under U. Given q, RootTrace returns a *suspect set* $S(q)$ (stage identifiers and/or artifact versions) and an *evidence bundle* $B(q)$ with which an auditor can independently check the trace behind $S(q)$ under the access policy.

Claim boundary. Unless additional domain constraints are available, RootTrace does not claim a unique root cause; it returns a verifiable candidate set. Section 6 therefore evaluates whether that candidate set is compact, auditable, and operationally useful.

3.2. Boundary, Actors, and Threat Model

The setting is multi-system and potentially multi-organization. We consider four roles:

- **Data providers**, producing source artifacts;
- **Settlement operator**, executing stages and publishing outputs;
- **Market participants**, raising or responding to discrepancy claims;
- **Auditor/verifier**, independently checking evidence bundles.

The traceability boundary starts when artifact/event records enter evidence recording and ends when $B(q)$ is verified. Manipulations entirely outside this boundary are not directly prevented unless they appear in recorded artifacts.

Within this boundary, we consider both benign and adversarial causes:

- **Benign causes**: data quality issues (missing/duplicate records), time alignment mismatches, and accidental misconfiguration or version drift.
- **Adversarial causes within scope**: unauthorized modification of recorded evidence, repudiation of actions (claiming a different version was used), silent deletion/rewriting of logs within the evidence recording boundary, and selective withholding of non-mandatory payload fields while still disclosing metadata needed for verification.

We assume non-fully-trusting parties, managed signing identities, and known trust anchors for verification. Raw payloads may be restricted, so commitments/metadata must support cross-party checking.

Out of scope. We do not claim semantic proof of settlement formulas, detection of attacks that leave no trace before evidence capture, or resolution of disputes that strictly require undisclosable payloads. Compromised trust anchors or full collusion of record producers/attesters are also outside primary scope.

Security goals are:

- **G1 (Integrity / tamper evidence).** Evidence records and their linkages should be tamper-evident such that unauthorized modifications are detectable.
- **G2 (Non-repudiation within stated identities).** Actors should not be able to plausibly deny recorded actions within the chosen identity model.

 doi:10.20944/preprints202604.1623.v1

- **G3 (Verifiability).** An auditor should be able to verify evidence bundles $B(q)$ under the defined disclosure policy.
- **G4 (Minimal disclosure).** Verification should require disclosing only necessary information; sensitive payloads may remain protected when compatible with the threat model.

These goals constrain design choices in Sections 5–6.

3.3. Running Example (Synthetic, for Illustration)

To ground the definitions without proprietary data, we use a compact synthetic example that is reused later only as a running reference.

Stages. For window W, three stages suffice: s_1 (*ingest*) materializes a metering snapshot; s_2 (*prepare*) aligns/aggregates to settlement granularity; s_3 (*settle*) computes an output from a rule specification and parameter set.

Artifacts. Data $D_m@v_{m1}$ (raw) and $D_a@v_{a1}$ (aligned); rule $R@v_{r1}$; parameters $P@v_{p1}$ vs. $P@v_{p2}$; jobs $J_1@v_{j1}$, $J_2@v_{j2}$, $J_3@v_{j3}$; outputs $O@v_{o1}$, $O@v_{o2}$ for the same scope and window. All $v_\cdot$ are content-addressed.

Events. e_1 at s_1 produces $D_m@v_{m1}$ under $J_1@v_{j1}$. e_2 at s_2 consumes $D_m@v_{m1}$ and emits $D_a@v_{a1}$ under $J_2@v_{j2}$. Reference event $e_3^{(ref)}$ at s_3 consumes $\{D_a@v_{a1}, R@v_{r1}, P@v_{p1}\}$ and emits $O@v_{o1}$ under $J_3@v_{j3}$; target event $e_3^{(tar)}$ differs only in $P@v_{p2}$ and output $O@v_{o2}$.

Injected cause. The only injected difference is parameter drift $P@v_{p1} \to P@v_{p2}$ at s_3, which induces a nonzero output difference Δ between $O@v_{o1}$ and $O@v_{o2}$.

Instantiated query. Following the query tuple $q = (W, \mathcal{U}, Y^{(ref)}, Y^{(tar)}, \Delta, \mathcal{C})$ in Section 3.1, the running example sets $Y^{(ref)} = O@v_{o1}$, $Y^{(tar)} = O@v_{o2}$, with $\mathcal{C}$ carrying run identifiers for the two outputs.

Expected interpretation. The two output traces differ only at the parameter version referenced by the settlement stage. The example therefore illustrates the type of localization target studied in the rest of the paper: a compact candidate set centered on the divergence stage/version, together with the evidence needed to justify that conclusion.

4. Path-Level Root Cause Localization of Settlement Discrepancies in Multi-Party Electricity Trading

4.1. RootTrace Overview

Multi-party electricity trading settlement involves multiple entities, stages, rule sets, and heterogeneous data sources. In this setting, discrepancies can be introduced by source-data bias, rule-execution inconsistency, parameter misconfiguration, temporal misalignment, or manual correction. Traditional item-by-item checking can detect a mismatch, but it is inefficient at isolating the true cause in a long dependency chain.

The core task is not discrepancy detection itself, but *path-level root-cause localization*: given an observed settlement discrepancy, identify where it was introduced and how it propagated across stages.

Inputs include trading declarations, metering data, settlement rules and parameters, intermediate results, final settlement outputs, and operation/correction logs. RootTrace returns three outputs: a root-cause candidate set, a ranking over those candidates, and explanation paths for discrepancy propagation.

This section describes the localization core of RootTrace and builds on the query model and boundary assumptions introduced earlier. The analysis proceeds through four linked steps: settlement-path modeling, discrepancy representation and propagation analysis, consistency-constrained diagnosis, and candidate generation with pruning and ranking.

At a coarse level, let the settlement process be represented as $P = (V, E)$, where V is the node set and E is the dependency-relation set. Each node $v_i \in V$ represents a processing unit (e.g., data-input node, rule-execution node, intermediate-result node, correction node, or output node), and carries an observed state value x_i. The terminal discrepancy is defined as $\Delta = y^{(obs)} - y^{(ref)}$, where $y^{(obs)}$ is the

 doi:10.20944/preprints202604.1623.v1

observed settlement result and $y^{(ref)}$ is the reference/theoretical result; vector-form Δ is used when settlement outputs are multi-dimensional. The overall RootTrace workflow is illustrated in Figure 1.

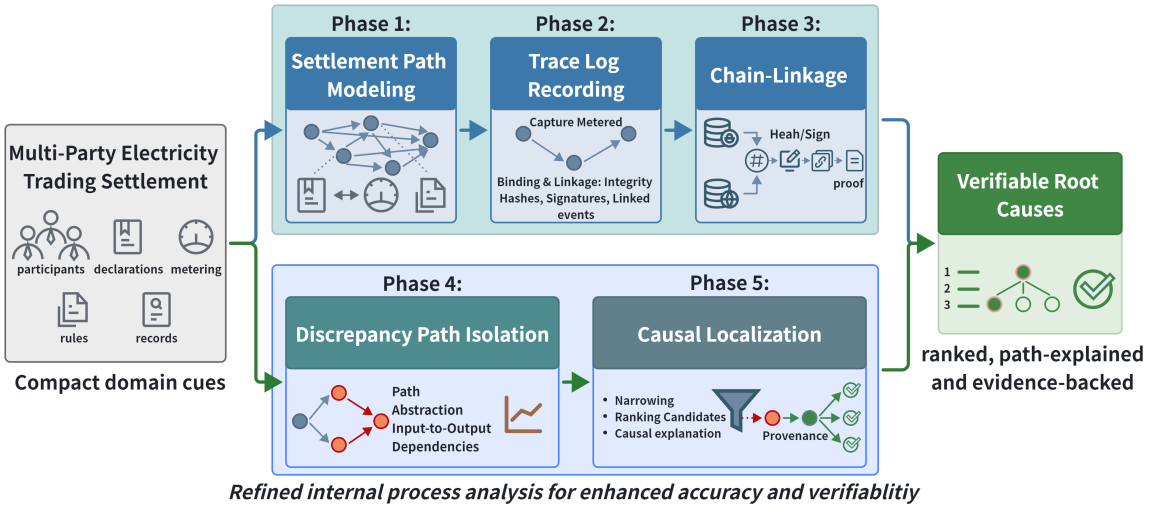


Figure 1. Overview of RootTrace.

4.2. Settlement Dependency Path Graph Construction

The settlement process in multi-party electricity trading is typically not linear. It is a coupled directed chain containing trading formation, data acquisition, rule matching, fee calculation, deviation correction, and bill generation. Although discrepancies are observed at output nodes, their true sources are often introduced in earlier stages. Therefore, we convert the business workflow into a structured dependency graph to support analyzable and traceable localization.

Node model. Nodes partition into data $\mathcal{V}_D$ (raw/intermediate quantities), rule $\mathcal{V}_R$ (rules, coefficients, parameters, charging standards), computation $\mathcal{V}_C$ (loss allocation, deviation correction, fees), correction $\mathcal{V}_M$ (manual review, overrides, backfill), and result $\mathcal{V}_O$ (bill lines and totals). Each v_i carries $\mathrm{Attr}(v_i) = \{\mathrm{type}_i, \mathrm{time}_i, \mathrm{source}_i, \mathrm{state}_i\}$.

Edge model. Dependencies are data flow (output feeds input), rule binding (computations tied to rule/parameter versions), temporal precedence, and aggregation (multiple upstreams joint into one result), written $e_{ij} = (v_i, v_j, r_{ij})$ with type r_{ij}.

Formal graph definition. We write the settlement dependency path graph as $G = (V, E, \Phi, \Psi)$, where V is the node set, E is the edge set, $\Phi : V \to \mathrm{Attr}$ maps node attributes, and $\Psi : E \to \mathcal{R}$ maps edge relation types. Under ordinary operation, G is topologically ordered and mostly acyclic. When manual rollback or review overwrite is present, write-back edges are allowed as controlled extensions. Figure 2 sketches an illustrative dependency-path graph for visualization.

The graph provides the structural basis for the remainder of the method section. Section 4.3 defines discrepancy propagation on G, Section 4.4 introduces consistency constraints for candidate reduction, and Section 4.5 gives the backward localization and ranking procedure used by RootTrace. Root-cause analysis is thus treated as structured inference over dependencies rather than as unrestricted search over logs. Section 5 later returns to the same graph from the verification side: the settlement dependency path graph is the localization-oriented view of the evidence graph, while signed records and append-only bindings supply the trust semantics needed to validate paths, versions, and claimed causes.

 doi:10.20944/preprints202604.1623.v1

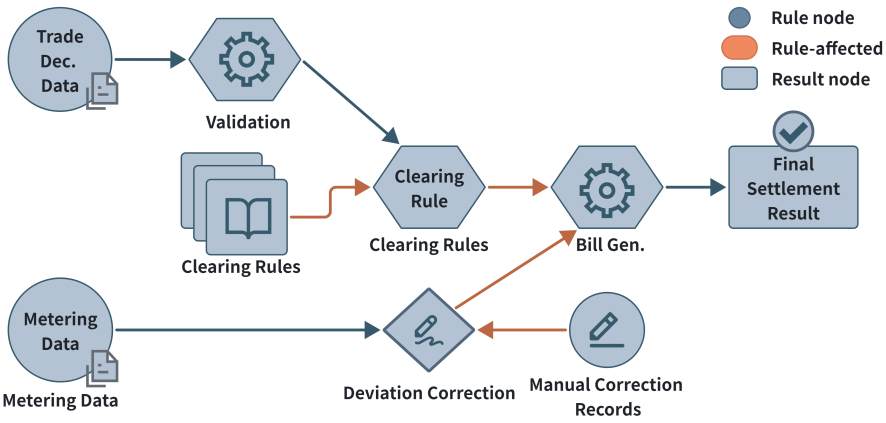


Figure 2. Settlement dependency-path graph.

4.3. Discrepancy Representation and Propagation Mechanism Modeling

Many localization approaches begin with backward search alone. Here, discrepancy is treated instead as a dynamic state on G, not as a static label. We call this formulation the Settlement Discrepancy Propagation Chain (SDPC).

Node-level discrepancy. For node v_i, numeric states use $\delta_i = x_i^{(obs)} - x_i^{(ref)}$. For non-numeric states (e.g., rule version, status flag), $\delta_i = 0$ if $x_i^{(obs)} = x_i^{(ref)}$ and $\delta_i = 1$ otherwise. Discrepancies are typed as numeric, rule, temporal, state, or aggregation.

General propagation rule. At node v_j, the discrepancy state is determined by discrepancies on its predecessors:

$$\delta_j = f_j\big(\{\delta_i \mid v_i \in \mathrm{Pa}(v_j)\}, \theta_j\big), \tag{1}$$

where $\mathrm{Pa}(v_j)$ is the predecessor set, θ_j denotes node-specific rule/parameter context, and $f_j(\cdot)$ is the propagation function.

Typical node-wise forms. Linear nodes combine predecessors as $\delta_j = \sum_{i \in \mathrm{Pa}(v_j)} w_{ij}\delta_i$; rule-selection nodes combine data and rule mismatch via $\delta_j = g(\delta_{data}, \delta_{rule})$; aggregators sum upstream effects plus bias, $\delta_j = \sum_{i=1}^{k} \delta_i + \epsilon_j$.

Key propagation mechanisms. In settlement chains, discrepancies may *accumulate* along long paths, *amplify* under aggregation or penalty scaling, or *selectively* propagate only when certain rules or triggers fire.

Under this formulation, terminal settlement anomalies are treated as a discrepancy-diffusion problem on G. This perspective motivates the backward candidate tracing defined next. Figure 3 illustrates a representative propagation path on such a graph.

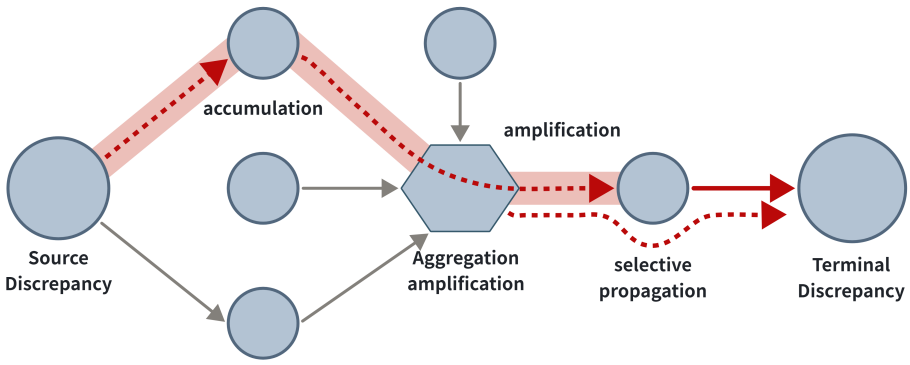


Figure 3. Discrepancy propagation along a settlement path.

 doi:10.20944/preprints202604.1623.v1

4.4. Anomaly Diagnosis with Consistency Constraints

Backward tracing on dependency paths alone usually leaves too many plausible suspects. Settlement pipelines also encode business consistency constraints that shrink that space in a principled way. We therefore fold consistency diagnosis into localization itself.

Constraint types. Four classes recur in settlement checks: **C1** mapped-field alignment $h(x_i, x_j) = 0$; **C2** effective rule/parameter identity $\text{rule}(v_j) = \text{rule}^*(t_j)$; **C3** temporal order $\text{time}(v_i) < \text{time}(v_j)$ on edges; **C4** aggregation balance $x_j = \sum_{v_i \in \text{Pa}(v_j)} x_i$.

Violation diagnosis. For each C_m, we use either a binary indicator $\Gamma(C_m) \in \{0,1\}$ (satisfied vs. violated) or a severity score $s_m = \text{ViolationScore}(C_m)$. In practice, the true cause often lies in subgraphs where violations recur or carry larger severity.

How constraints guide localization. Constraints prune impossible candidates, highlight conflict-centric subgraphs, and improve ranking. For node v_i, we write $\text{Score}_c(v_i) = \sum_{C_m \in \mathcal{C}(v_i)} \alpha_m s_m$, where $\mathcal{C}(v_i)$ are the constraints touching v_i and α_m are their weights. Figure 4 depicts constraint-guided diagnosis in schematic form.

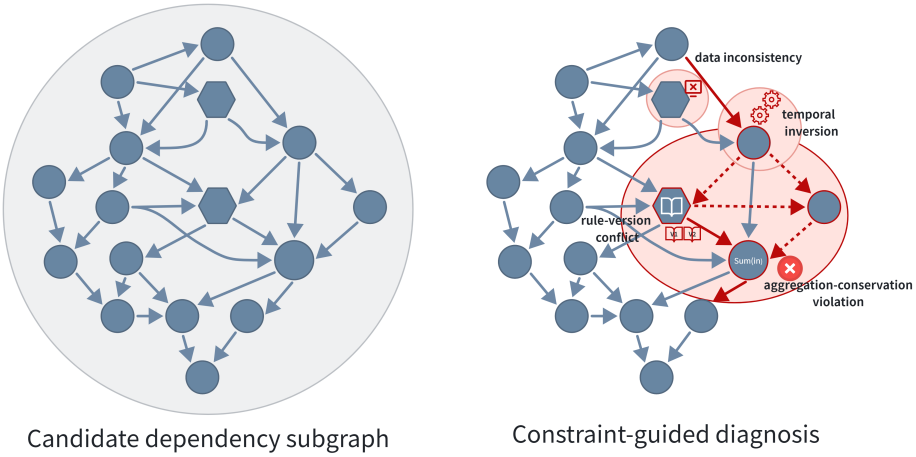


Figure 4. Constraint-guided discrepancy diagnosis.

4.5. Root Cause Candidate Generation and Localization Algorithm

For $q = (W, U, Y^{(ref)}, Y^{(tar)}, \Delta, \mathcal{C})$, the algorithm returns ranked root-cause candidates together with explanation-ready paths.

4.5.1. Discrepancy-Impacted Node Identification

Starting from the terminal discrepancy outputs, we build the discrepancy-impacted subgraph $G_\Delta = (V_\Delta, E_\Delta)$. The node set V_Δ contains nodes with $\delta_i \neq 0$ as well as nodes that remain path-connected to discrepancy nodes under SDPC-valid reverse traversal. The search therefore proceeds backward from the terminals while preserving propagation consistency, which confines localization to the discrepancy-relevant region of the graph.

4.5.2. Backward Candidate Tracing

Within G_Δ, backward tracing yields a baseline pool $R = \{v_i \in V_\Delta \mid \delta_i \neq 0,\ \text{Pred}(v_i) \text{ is locally normal}\}$, meaning that the predecessors do not exhibit equally strong discrepancy or conflict under the same boundary. Define $\text{Anom}(v_i) = \beta_1 \widetilde{\delta}_i + \beta_2 \widetilde{\text{Score}}_c(v_i)$ with normalized $\widetilde{\delta}_i, \widetilde{\text{Score}}_c(v_i)$ and $\beta_1 + \beta_2 = 1$. A predecessor set is locally normal when $\max_{v_j \in \text{Pa}(v_i)} \text{Anom}(v_j) < \tau_{\text{pred}}$ for threshold τ_{pred}. The pool is then expanded using earliest-abnormal, high-conflict, and high-propagation source patterns, with SDPC-valid path evidence stored for each candidate.

 doi:10.20944/preprints202604.1623.v1

4.5.3. Candidate Pruning and Scoring

Candidates are pruned by boundary and feasibility checks, then scored by propagation contribution $\text{Score}_p(v_i)$, constraint conflict $\text{Score}_c(v_i)$ (Section 4.4), and terminal coverage $\text{Score}_h(v_i)$. For candidate v_i, let $\Pi(v_i)$ denote the SDPC-valid paths from v_i to terminal discrepancy nodes and $w(e) \in [0,1]$ the normalized edge weights. The propagation score $\text{Score}_p(v_i) = \sum_{\pi \in \Pi(v_i)} \big(\prod_{e \in \pi} w(e)\big)\rho(\pi)$ weights each path by $\rho(\pi) \in [0,1]$, which captures consistency with the observed discrepancy types and triggered constraints. With impacted terminals $\mathcal{T}_\Delta$ and $\mathcal{T}(v_i) \subseteq \mathcal{T}_\Delta$ reachable from v_i, coverage is $\text{Score}_h(v_i) = |\mathcal{T}(v_i)|/|\mathcal{T}_\Delta|$.

The final ranking score is

$$\text{Score}(v_i) = \lambda_1 \text{Score}_p(v_i) + \lambda_2 \text{Score}_c(v_i) + \lambda_3 \text{Score}_h(v_i), \tag{2}$$

with $\lambda_1, \lambda_2, \lambda_3 \geq 0$, $\sum \lambda_i = 1$ (set empirically or by validation). Output $S(q) = \text{TopK}(\text{Score}, R)$.

Proposition 1 (Candidate inclusion under observable single-root-cause conditions). Under a single-root-cause assumption, complete observability of the relevant dependency subgraph, and correctly specified SDPC-valid propagation and constraint rules, the true root-cause node v^* belongs to the candidate pool R or to its source-pattern expansion.

Proof sketch. If v^* is the earliest node with nonzero discrepancy inside the boundary, then $\max_{v_j \in \text{Pa}(v^*)} \text{Anom}(v_j) < \tau_{\text{pred}}$; otherwise an earlier predecessor would already appear abnormally strong. Because $\delta_{v^*} \neq 0$ and $v^* \in V_\Delta$, v^* satisfies the construction rule for R or is retained by source-pattern expansion. It is therefore not removed before ranking. The claim is recall-oriented, not a uniqueness result.

4.5.4. Localization Output and Explanation

The output is a structured Top-k set $O = \{(v_i, \text{Score}(v_i), \text{Path}_i, \text{Explain}_i)\}_{i=1}^{k}$, where Path_i records the propagation path and Explain_i contains the associated constraint evidence and audit notes.

Algorithm 1. Path-level root-cause localization in RootTrace.

Input: settlement dependency graph G, discrepancy observations, constraint set $\mathcal{C} = \{C_1, \ldots, C_4\}$
Output: ranked root-cause candidates $S(q)$ and Top-k explanation tuples O
Construct discrepancy-impacted subgraph $G_\Delta = (V_\Delta, E_\Delta)$ by reverse traversal from terminal discrepancy nodes
For each $v_i \in V_\Delta$, compute $\text{Score}_c(v_i)$ and local anomaly strength $\text{Anom}(v_i)$
Build initial candidate pool R: nodes with $\delta_i \neq 0$ and $\max_{v_j \in \text{Pa}(v_i)} \text{Anom}(v_j) < \tau_{\text{pred}}$
Expand R with earliest-abnormal, high-conflict, and high-propagation source patterns; record SDPC-valid paths $\Pi(v_i)$ for each $v_i \in R$
for each candidate $v_i \in R$ **do**
    Compute $\text{Score}_p(v_i) = \sum_{\pi \in \Pi(v_i)} \big(\prod_{e \in \pi} w(e)\big)\rho(\pi)$
    Compute $\text{Score}_h(v_i) = |\mathcal{T}(v_i)|/|\mathcal{T}_\Delta|$
    Compute $\text{Score}(v_i) = \lambda_1 \text{Score}_p(v_i) + \lambda_2 \text{Score}_c(v_i) + \lambda_3 \text{Score}_h(v_i)$
end for
Rank candidates by $\text{Score}(v_i)$, output $S(q)$, and assemble Top-k tuples O

---

 doi:10.20944/preprints202604.1623.v1

Table 2. Key notation for RootTrace localization and ranking.

Term	Meaning
$\mathrm{Anom}(v_i)$	Local anomaly strength used to decide whether v_i enters the initial candidate pool.
β_1, β_2	Mixture weights for discrepancy magnitude and normalized constraint conflict in $\mathrm{Anom}(v_i)$, with $\beta_1 + \beta_2 = 1$.
τ_{pred}	Predecessor-normality threshold used when seeding the baseline candidate pool.
$\mathrm{Score}_c(v_i)$	Weighted constraint-conflict score over the C1–C4 checks incident on v_i.
$\mathrm{Score}_p(v_i), \rho(\pi)$	Path-propagation score and path-consistency factor over SDPC-valid paths $\pi \in \Pi(v_i)$.
$\mathrm{Score}_h(v_i)$	Terminal-coverage ratio $\lvert\mathcal{T}(v_i)\rvert / \lvert\mathcal{T}_\Delta\rvert$.
$\lambda_1, \lambda_2, \lambda_3$	Convex weights in the final ranking score of Eq. 2.
Top$K$	Cutoff used when converting scored candidates into the reported suspect set $S(q)$.

4.6. Method Complexity and Interpretability Analysis

4.6.1. Complexity Analysis

Let $|V|$, $|E|$ be the graph size, $|V_\Delta|, |E_\Delta|$ the impacted subgraph, $|\mathcal{C}|$ the evaluated constraints, and $|R|$ the candidates. Graph construction costs $O(|V| + |E|)$; impacted-subgraph search $O(|V_\Delta| + |E_\Delta|)$; constraint checks $O(|\mathcal{C}|)$ under bounded per-constraint work; scoring is $O(|R|)$ and sorting $O(|R| \log |R|)$. Overall $O(|V| + |E| + |V_\Delta| + |E_\Delta| + |\mathcal{C}| + |R| \log |R|)$, which behaves near-linearly on sparse graphs with moderate $|R|$.

4.6.2. Interpretability Analysis

Interpretability comes from explicit structure rather than from a single opaque score. The graph shows where effects may flow, SDPC rules explain how discrepancies combine, constraints justify pruning and retention, and the exported tuples preserve paths together with the evidence needed for review and audit.

4.6.3. Robustness and Applicability Boundary

Partially missing intermediates still allow approximate localization from local paths and visible constraints; multiple concurrent causes are handled by Top-k outputs rather than a forced singleton; and when reference values are missing, replayed rule outputs or historical baselines can stand in. Limits are equally clear: quality depends on how much structure and linkage can be extracted; very sparse logs or absent intermediates weaken confidence; and tight closed-loop workflows may need richer dynamic path models than those assumed here.

5. Trustworthy Evidence Recording and Verification for Root Cause Localization

5.1. Design Goals, Threat Scope, and Role in Localization

Section 4 addressed how RootTrace localizes candidate causes along settlement dependency paths. A separate question remains: why should such a localization claim be trusted? Each claimed path, candidate, and responsibility statement must be grounded in evidence records that another party can verify independently.

The mechanism is organized around four goals: integrity and tamper evidence, identity-bound accountability, independent verifiability, and minimal disclosure. Its scope is equally explicit: claims are supported only *inside the evidence recording boundary*, not for unrecorded real-world events.

Trusted evidence boundary. Three layers matter. *Outside* the recorder lies physical reality that has not yet been captured. *Inside* the recording boundary sit signed, canonicalized records and append-only log order. *Localization claims* extend only as far as those records and their verified dependencies justify. A verified trace is therefore not the same as unrestricted proof about unrecorded events.

5.2. Verifiable Evidence Model

This subsection does not introduce a second analytical object. It specifies the minimum evidence semantics required to validate the localization outputs of Section 4. The same path structure used for localization is also the one checked by an independent verifier. Two record types suffice:

 doi:10.20944/preprints202604.1623.v1

artifact-version records r_A (artifact and version IDs, scope such as W, U, producer identity, optional payload commitment) and **event** records r_E (event ID/type, actor, time, input/output version IDs, referenced rule/parameter versions). They map directly onto the localization model: versions become data/version nodes, events become transform nodes, I/O links become dependency edges, and referenced rule/parameter fields become rule edges. Verification thus inspects the same object on which localization operates, rather than rebuilding the problem on a parallel graph.

Time and version binding are part of the same evidence model: each record binds event time and exact version identifiers so that discrepancy localization can be replayed on identical snapshots and rule sets.

5.3. Integrity and Non-Repudiation Bindings

To make evidence tamper-evident and accountable, each record is canonicalized, hashed, identity-signed, and inserted into an append-only log. The canonicalization policy follows a few fixed principles: records are serialized as fixed-field objects with explicit field names; all identifiers remain case-sensitive and are not alias-normalized during verification; input/output/reference lists are sorted deterministically before serialization; timestamps and window identifiers use one normalized format; missing optional fields are represented explicitly rather than omitted ad hoc; payload commitments always carry their algorithm identifier; and digests are computed only over canonical serialized bytes, excluding detached signatures and transport wrapper fields.

Cryptographically, each record binds *content* via $h(r)$ on canonical bytes (or commitment payload), *identity* via signatures on $h(r)$ under the governance key policy, and *append-only order* via hash-chained log entries so deletion or reordering surfaces. Storage may be WORM, append-only databases, or permissioned ledgers, any backend preserving those checks suffices.

5.4. Evidence Bundle Extraction and Verification

Given a discrepancy query q and the localization output from Section 4, the system derives a claimed provenance slice and exports an evidence bundle $B(q)$. The bundle includes the records, signatures, and log-linkage material required for independent replay and checking.

Candidate nodes and explanation paths are packaged into a claimed provenance slice and an exportable bundle $B(q)$. The bundle carries the records, signatures, version references, and log-linkage material needed to justify the localization result within the stated trust boundary. Independent verification then checks whether the claimed slice is structurally complete and cryptographically trustworthy; that outcome in turn calibrates the credibility, granularity, and accountability of the final localization claim. Concretely, an auditor verifies $B(q)$ through four checks: (V1) record integrity, (V2) signature validity, (V3) append-only consistency, and (V4) trace consistency.

Verification procedure. Algorithm 2 outlines a reference procedure independent of storage backend.

 doi:10.20944/preprints202604.1623.v1

Algorithm 2. Evidence-bundle verification.

Input: evidence bundle $B(q)$, trusted public keys / trust anchors, optional published log checkpoint
Parse $B(q)$ into: record set $\mathcal{R}$, signature set Σ, log linkage material $\mathcal{L}$, and claimed trace subgraph G_B
for all record $r \in \mathcal{R}$ **do**
  Canonicalize r and compute digest $h(r)$
  Check that $h(r)$ matches the digest/commitment included in r (if present)
end for
for all signature $\sigma \in \Sigma$ **do**
  Verify σ against the claimed signer identity and trusted public key material
end for
Verify append-only consistency using $\mathcal{L}$
Verify trace consistency of G_B via event input/output references
Output: ACCEPT if all checks pass; otherwise REJECT with a minimal failure report

Minimal failure report. On reject, the verifier returns a compact triple (error class, affected object identifier, details). Representative classes include canonicalization failure; digest or commitment mismatch; invalid signature or unknown key; broken append-only linkage; missing referenced record; inconsistency between the claimed trace and record references; and insufficient disclosure for the requested verification task. These failures affect localization credibility differently. A digest mismatch weakens the affected node and its downstream path as strong evidence; a signature failure preserves object existence but downgrades responsibility attribution; a broken append-only chain weakens time-order and completeness claims; missing references prevent full reverse traceback and force partial or stage-level localization; and insufficient disclosure reduces localization granularity from fine-grained root cause to coarse object- or stage-level diagnosis.

5.5. Verification Limits and Impact on Localization

Cross-organization investigations require balancing verifiability and confidentiality. We keep a minimal-disclosure policy: reveal identifiers, versions, signatures, and log-linkage fields needed for verification, while allowing sensitive payloads to remain protected behind access control with commitments. The disclosure rule is simple: identifiers needed to reconstruct graph structure must remain in cleartext, including artifact IDs, version IDs, event IDs, event/stage types, signer identity, event time, and window binding. Payload contents such as raw data and proprietary rule text may remain protected, but their commitments must be disclosed. Optional metadata such as scope hints may be disclosed when it improves filtering granularity and does not violate policy. Verification limits must also be explicit: if key linkage fields are hidden, independent path validation cannot be completed; if payload semantics are hidden, value-level discrepancy interpretation may be limited even when structural trace validation succeeds. Verification feeds back into Section 4: passing V1–V4 makes candidates and paths evidence-backed; integrity or chain failures demote or drop affected paths; signature failures may preserve structural explanation while weakening actor attribution; and insufficient disclosure caps localization at the finest level visible fields allow. Section 5 is therefore not a parallel "method track" but the layer that calibrates confidence, granularity, and accountability for Section 4 outputs.

6. Evaluation

This section evaluates RootTrace from three complementary perspectives: localization effectiveness, verification effectiveness, and operational overhead. The focus is whether RootTrace localizes settlement discrepancies more accurately and with less investigation effort than weaker tracing baselines, and whether the exported evidence bundle makes those conclusions independently verifiable

 doi:10.20944/preprints202604.1623.v1

under realistic disclosure constraints. Tables 3–6 and Figs. 5–12 report the quantitative outcomes, covering the primary benchmark, trusted-verification analysis, ablation and overhead studies, the unattended stress run, and one end-to-end case study.

6.1. Experimental Setup

The evaluation addresses four questions: localization accuracy, analyst effort, verification and tamper detection, and component contribution. All experiments use one offline simulator implementing the artifact–event schema of Section 3, with dependency graphs materialized in memory and all methods run under the same execution protocol.

Implementation and hardware

The implementation is in Python on Linux. Record digests use SHA-256; digital signatures, when enabled, use Ed25519. B5 uses a lightweight ridge-regularized graph scorer in the spirit of learned node scoring on dependency graphs [18], with fixed coefficients and no GPU dependence. Timings were obtained on a commodity workstation.

Public reference data (provenance)

The benchmark is semi-synthetic. Workflow templates are motivated by published wholesale-market and integration studies [4–6,17] and instantiated into labeled cases by varying participants, windows, rule versions, and correction events. Each case follows the artifact-version-event schema of Section 3. A standardized monthly macroeconomic panel (FRED-MD) [19] is attached only as external covariates and does not define ground truth. The main grid covers six discrepancy families (raw-data, rule-version, temporal, transformation, correction, and evidence) across *small*, *medium*, and *large* templates under a linear topology. The scoring hyperparameters $\beta_1, \beta_2, \tau_{\text{pred}}, \lambda_1, \lambda_2, \lambda_3$, and Top$K$ are selected offline and then frozen for all reported runs.

The baselines form an incremental ladder on the same graph and logging interface:

- **B1:** rule-based discrepancy checking.
- **B2:** stage-order backtracking without graph reasoning.
- **B3:** graph backtracking without constraints or SDPC scoring.
- **B4a:** graph backtracking with constraints but without trust support.
- **B4b:** SDPC-style localization without trusted-bundle semantics.
- **B5:** a supplementary lightweight graph-learning scorer [18].

This progression isolates the contributions of graph structure, constraints, SDPC propagation, and trusted evidence handling.

The metrics fall into three groups:

- **Localization metrics:** Top-1, Top-3, Top-5, MRR, graph distance to ground truth, and stage-level accuracy.
- **Explainability and analyst-effort metrics:** candidate-set size, reduction ratio, first-hit rank, path completeness, and explainability-node graph coverage when reported.
- **Trusted-traceability metrics:** tamper detection, clean-bundle verification success, false rejection, verification wall time, record-ingestion latency, and serialized bundle size.

Clean bundles are further subjected to controlled tamper cases after export, including digest mismatch, missing reference, broken append-only chain, timestamp inconsistency, and insufficient disclosure; supplementary stress cases add wrong-signature and path-hallucination tampering. The ablation study removes path scoring, discrepancy propagation, constraints, and trust one at a time. Unless otherwise noted, all reported values are mean $\pm$ standard deviation over **ten** independent repetitions with distinct global random seeds.

 doi:10.20944/preprints202604.1623.v1

Reproducibility

The simulator specification, benchmark grid, frozen hyperparameters, and random-seed protocol are stated here at a level intended to support independent reproduction of the reported tables and figures without relying on repository layout.

6.2. Experimental Results and Analysis

The results are organized into four primary groups, one unattended stress study, and one case study, following Section 6.1. Tables 3–5 report mean ± std over ten independent full-grid replications. Figures 5–11 use the same replication policy, Figure 10 summarizes a separate eight-hour unattended stress run, and Figure 12 presents one representative instance.

6.2.1. Localization Effectiveness and Analyst Effort

The primary comparison is between RootTrace and baselines B1–B4b, with B5 included as a supplementary learned comparator under the same averaging protocol [14,18].

Under the frozen hyperparameters of Section 6.1, RootTrace and B4b coincide on the localization columns of Table 3. This is expected, since B4b retains the same SDPC ranking and constraint machinery while removing trusted-bundle semantics only. B5 achieves the strongest shortlist behavior on this grid, especially in Top-3, Top-5, and path completeness, whereas B1–B4a illustrate the familiar trade-off between aggressive narrowing and structurally faithful explanation.

Figures 5 and 6 unpack these aggregate trends. The family-wise panels show that the relative gains vary by discrepancy type, with rule-version, temporal, correction, and evidence cases benefiting more from structured path reasoning than raw-data-only cases. The explainability view separates shortlist reduction from explanation fidelity, making clear that methods without exported paths can still narrow the candidate set but do not provide the same level of auditable path support.

Table 3. Localization quality and analyst effort for RootTrace and baselines. Values are mean ± std over ten replications.

Method	Top-1	Top-3	Top-5	MRR	Dist↓	Stage Acc.	Cand.Size↓	Red.Ratio↑	Inspect↓	PathComp↑
RootTrace	0.472 ± 0.029	0.650 ± 0.046	0.833 ± 0.000	0.598 ± 0.019	2.61 ± 0.18	0.472 ± 0.029	4.44 ± 0.00	0.400 ± 0.000	2.07 ± 0.10	0.379 ± 0.011
B1 Rule-based	0.167 ± 0.000	0.167 ± 0.000	0.167 ± 0.000	0.167 ± 0.000	4.00 ± 0.00	0.333 ± 0.000	1.00 ± 0.00	0.965 ± 0.000	1.00 ± 0.00	0.000 ± 0.000
B2 Stage-order	0.039 ± 0.037	0.117 ± 0.049	0.194 ± 0.047	0.089 ± 0.038	3.72 ± 0.20	0.167 ± 0.000	5.00 ± 0.00	0.824 ± 0.000	4.60 ± 0.16	0.000 ± 0.000
B3 Graph-no-constraints	0.167 ± 0.000	0.500 ± 0.000	0.628 ± 0.079	0.334 ± 0.019	2.58 ± 0.04	0.256 ± 0.039	5.00 ± 0.00	0.824 ± 0.000	3.43 ± 0.07	0.000 ± 0.000
B4a Graph-constraints-no-trust	0.256 ± 0.039	0.500 ± 0.000	0.617 ± 0.081	0.374 ± 0.025	2.82 ± 0.08	0.344 ± 0.078	5.00 ± 0.00	0.824 ± 0.000	3.39 ± 0.05	0.000 ± 0.000
B4b SDPC-no-trusted-bundle	0.472 ± 0.029	0.650 ± 0.046	0.833 ± 0.000	0.598 ± 0.019	2.61 ± 0.18	0.472 ± 0.029	4.44 ± 0.00	0.400 ± 0.000	2.07 ± 0.10	0.379 ± 0.011
B5 Graph-learning (supplementary)	0.278 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.574 ± 0.000	3.17 ± 0.00	0.389 ± 0.000	5.00 ± 0.00	0.824 ± 0.000	2.11 ± 0.00	0.667 ± 0.000

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 April 2026 doi:10.20944/preprints202604.1623.v1

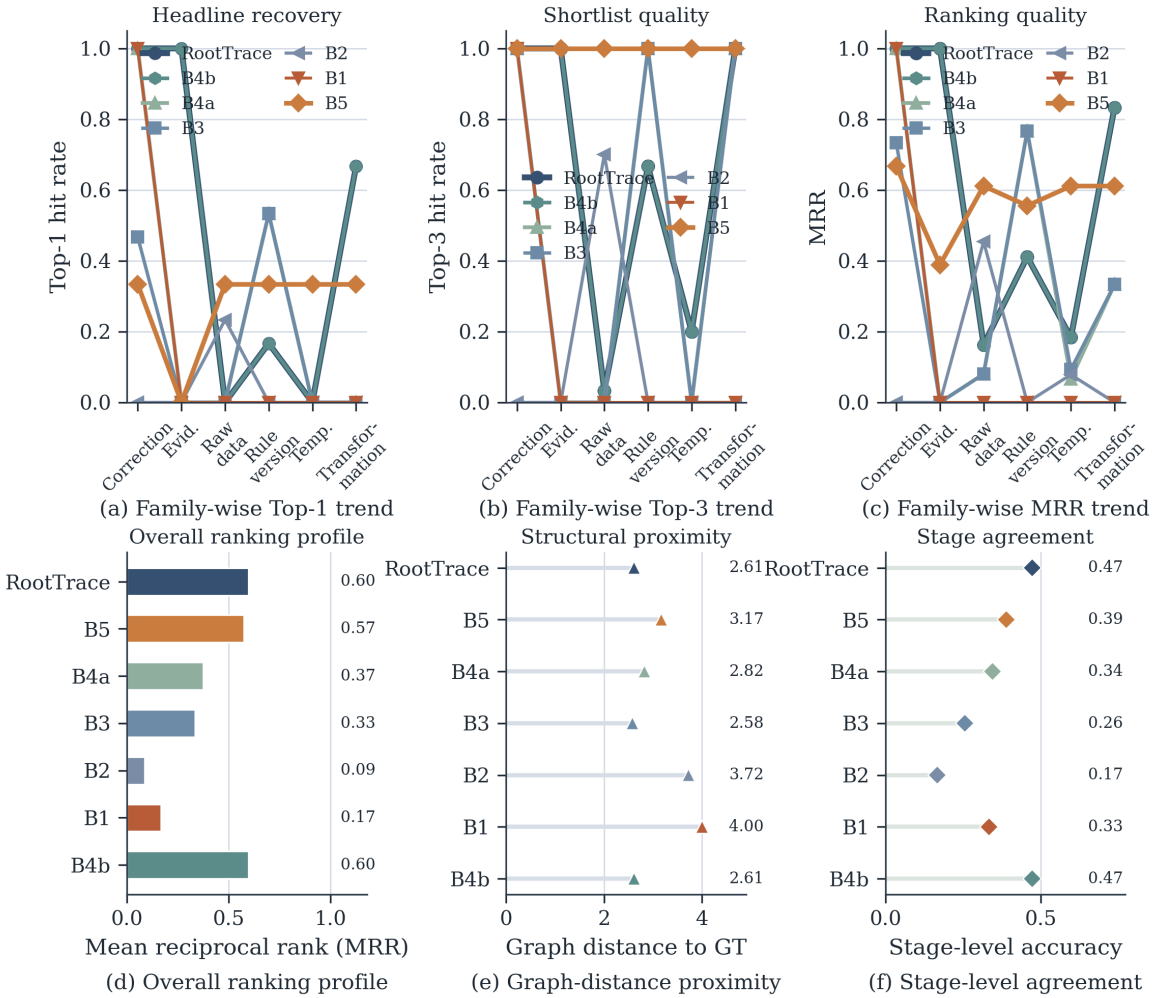


Figure 5. Grouped localization results for the primary benchmark.

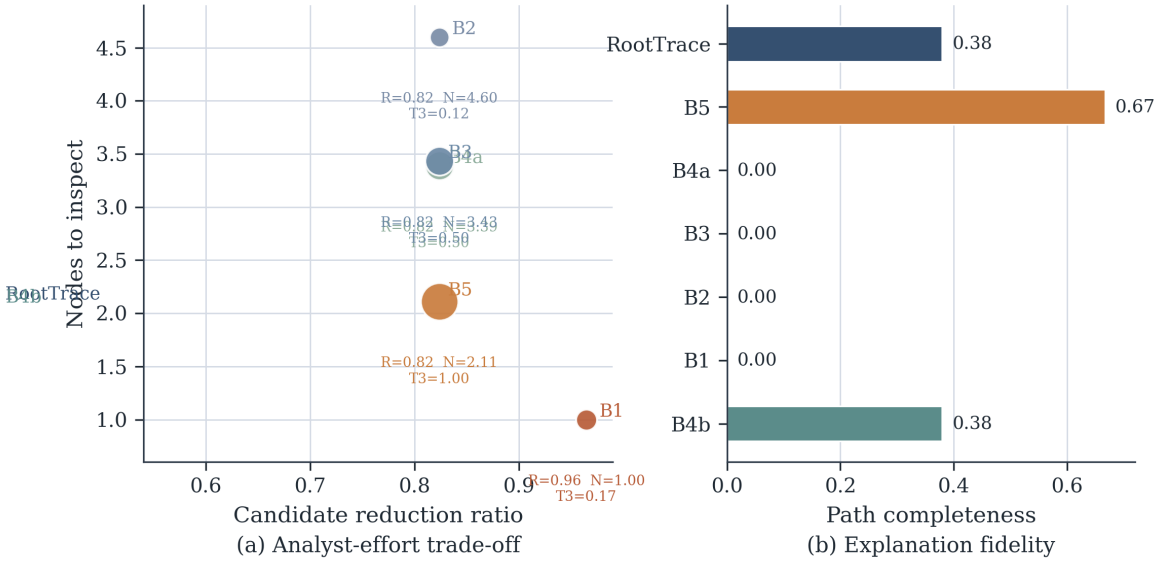


Figure 6. Explainability and analyst-effort results.

Figure 7 complements the headline metrics with structural diagnostics. In particular, first-hit rank, graph distance, and stage accuracy do not always move with exact Top-1 recovery, which is why

 doi:10.20944/preprints202604.1623.v1

the grouped view is useful for distinguishing shortlist quality from structural proximity to the true cause region.

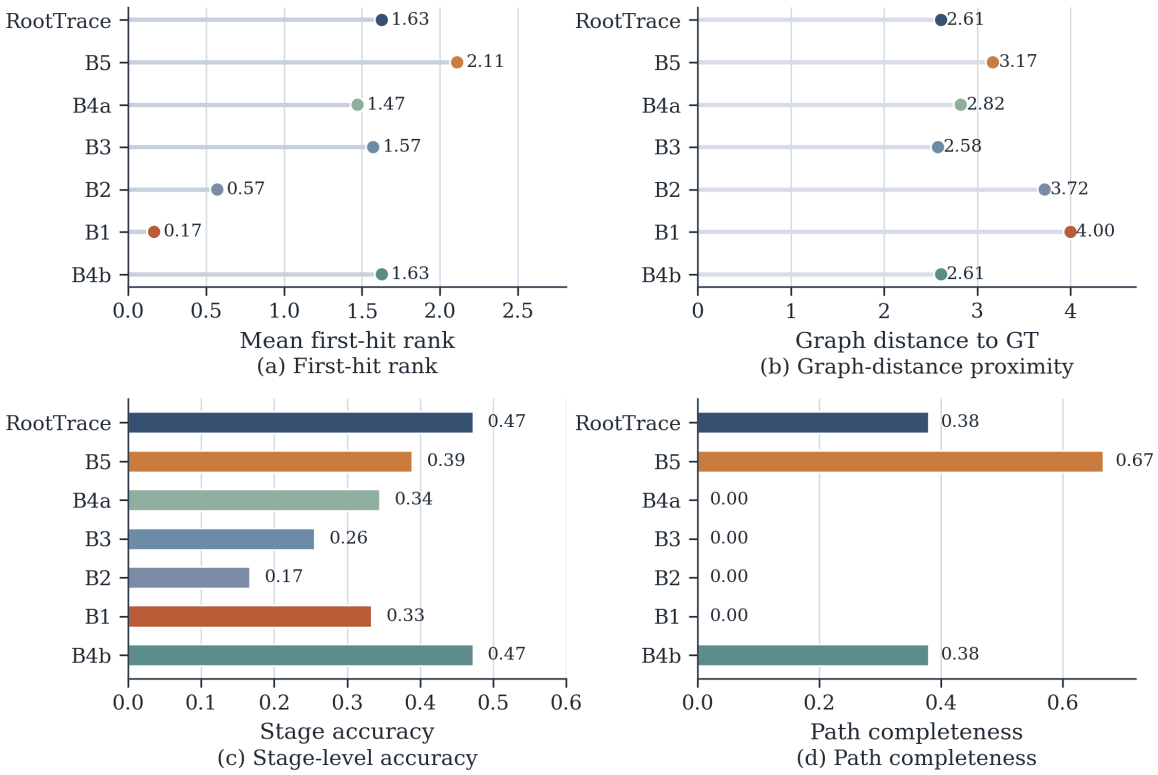


Figure 7. Ranking and structural diagnostics.

6.2.2. Trusted Verification and Tamper Detection

The verification study covers clean-bundle acceptance and tamper detection for the classes defined in Section 6.1: digest mismatch, missing reference, broken chain, timestamp inconsistency, insufficient disclosure, and the supplementary extended classes.

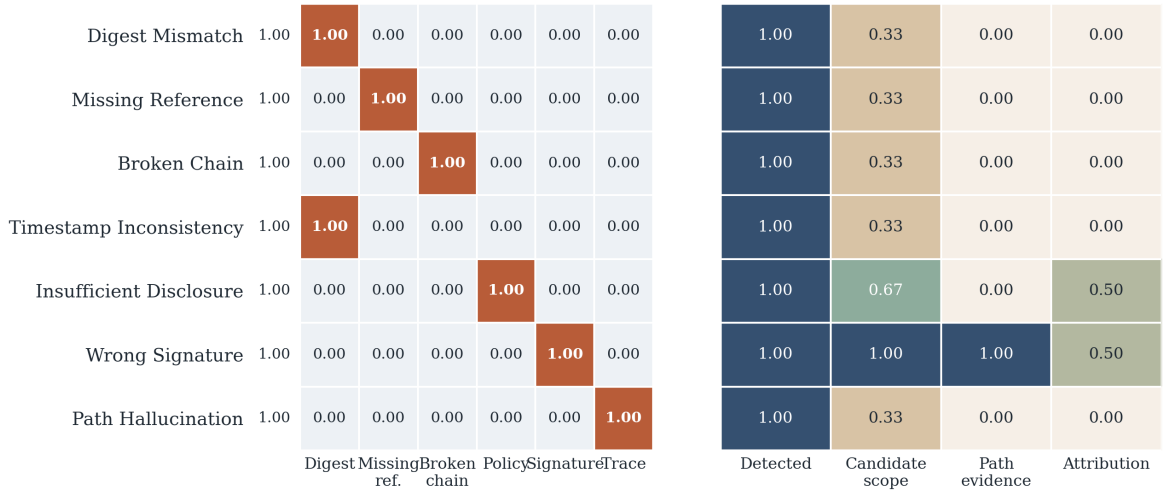
Table 4 shows stable clean-bundle acceptance, zero measured false rejection on clean bundles, and full detection across the configured tamper classes under the reference checker. Since aggregate detection is saturated on this grid, Figure 8 is more informative than the rate table alone: different tamper classes weaken different parts of the final claim, with structural faults primarily degrading path support and disclosure or signature faults limiting granularity or attribution.

Table 5 and Figure 9 show that discrepancy propagation contributes most strongly to ranking quality on this benchmark, while removing trust mainly affects tamper detection rather than localization. Removing constraints changes the candidate geometry more aggressively, yielding a smaller pool but a different ranking regime. Taken together, the ablations support the interpretation that propagation, constraints, and trust contribute through different mechanisms rather than through one interchangeable score term.

 doi:10.20944/preprints202604.1623.v1

Table 4. Verification and trusted-traceability results for RootTrace bundles.

Metric	Value	Notes
Clean verification success rate	1.000 ± 0.000	on untampered bundles
False rejection rate	0.000 ± 0.000	clean bundles incorrectly rejected
Tamper detection rate	1.000 ± 0.000	aggregated over all tamper types
Digest mismatch detection	1.000 ± 0.000	tamper subtype
Missing reference detection	1.000 ± 0.000	tamper subtype
Broken chain detection	1.000 ± 0.000	tamper subtype
Timestamp inconsistency detection	1.000 ± 0.000	tamper subtype
Insufficient disclosure detection	1.000 ± 0.000	tamper subtype
Wrong signature detection (supplementary)	1.000 ± 0.000	tamper subtype
Path hallucination detection (supplementary)	1.000 ± 0.000	tamper subtype
Graph construction latency (ms)	0.164 ± 0.019	dependency graph build only
Bundle materialization latency (ms)	1.804 ± 0.058	export bundle only
Mean verification latency (ms)	4.34 ± 0.12	per bundle
Record-ingestion latency (ms)	1.97 ± 0.06	graph build + bundle export (total)
Storage overhead (bytes/bundle)	34476 ± 48	serialized bundle size



(a) Dominant verifier code (b) Retained localization claim

Figure 8. Verification semantics by tamper class.

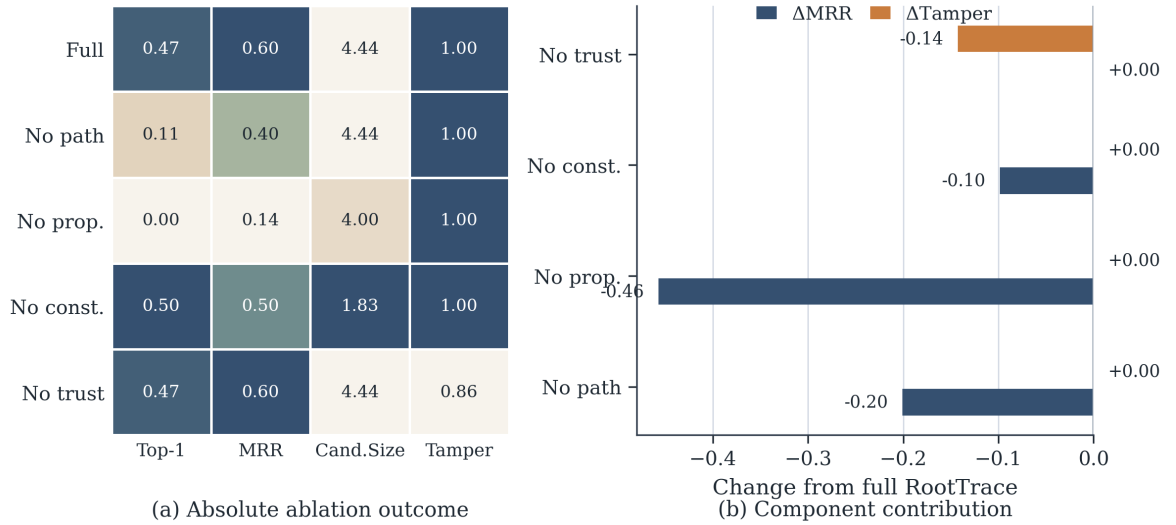
6.2.3. Ablation and Scale Overhead

The ablation study isolates the contribution of path modeling, discrepancy propagation, consistency constraints, and trusted recording.

Table 5. Ablation results for RootTrace. Each row removes one component from the full configuration.

Variant	Top-1	MRR	Cand.Size↓	Tamper Detect.
RootTrace	0.472 ± 0.029	0.598 ± 0.019	4.44 ± 0.00	1.000 ± 0.000
No path scores	0.111 ± 0.000	0.398 ± 0.012	4.44 ± 0.00	1.000 ± 0.000
No discrepancy propagation	0.000 ± 0.000	0.141 ± 0.009	4.00 ± 0.00	1.000 ± 0.000
No constraints	0.500 ± 0.000	0.500 ± 0.000	1.83 ± 0.00	1.000 ± 0.000
No trust	0.472 ± 0.029	0.598 ± 0.019	4.44 ± 0.00	0.857 ± 0.000

 doi:10.20944/preprints202604.1623.v1



(a) Absolute ablation outcome (b) Component contribution

Figure 9. Grouped ablation results for RootTrace.

6.2.4. Long-Run Autonomous Stress Log

An additional **eight-hour** unattended loop cycles over scales, discrepancy families, and topologies, executes SDPC localization plus signed-bundle verification on a per-iteration subset of tamper classes, and adapts localization hyperparameters online against fixed joint thresholds on localization and verification. The design intentionally stresses the controller, so many iterations trigger parameter updates rather than satisfying the joint target immediately. No runtime failures occurred, and Table 6 reports the aggregate counts. The terminal hyperparameters ($\tau_{\text{pred}} = 0.25$, λ_1=0.55, λ_2=λ_3=0.22 after adaptation) differ from the fixed settings used for Tables 3–5; the stress log therefore should not be read as numerically identical to the primary ten-repeat tables.

Table 6. Eight-hour unattended stress study.

Metric	Value
Wall-clock duration limit	8 hours
Completed iterations	1,330,578
Threshold-satisfying iterations	147,841 (≈11.1%)
Iterations requiring retuning	1,182,737
Runtime-failure iterations	0

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 April 2026 doi:10.20944/preprints202604.1623.v1

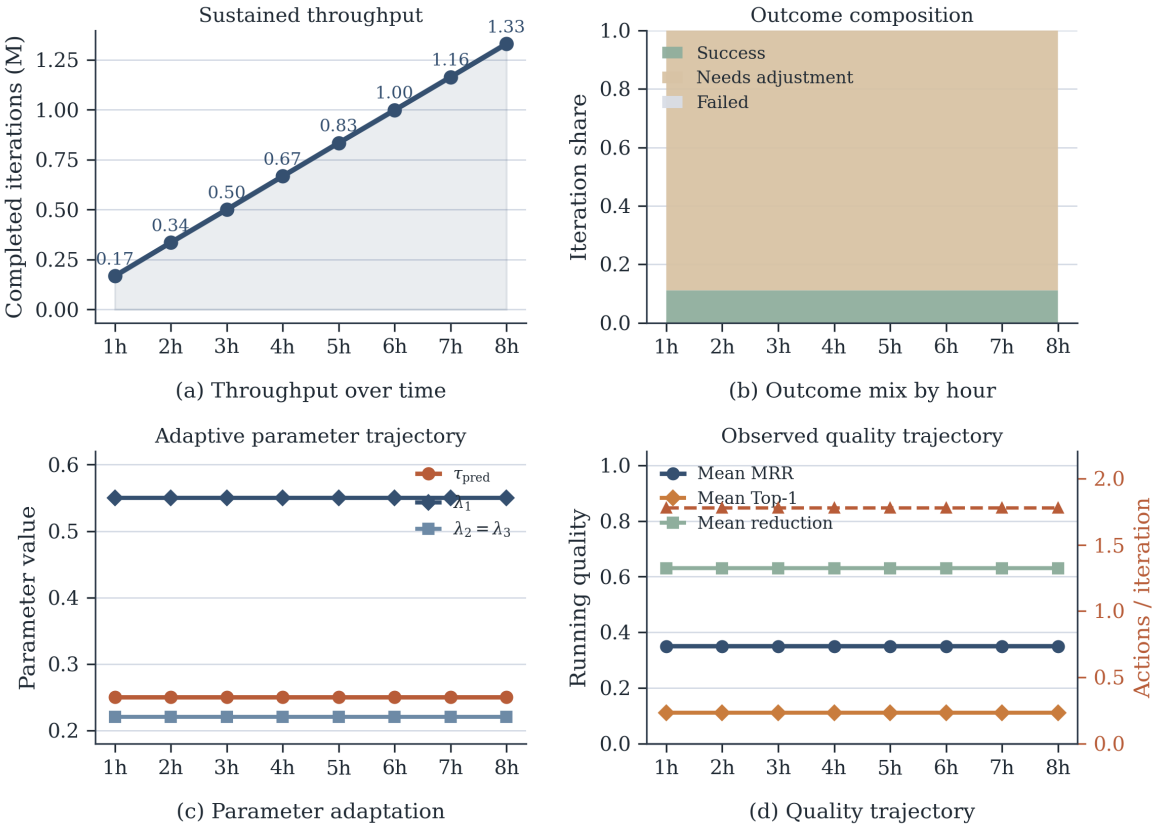


Figure 10. Autonomous-stress trends over the eight-hour unattended run.

Figure 10 complements Table 6 by showing how the unattended loop behaves over time rather than only at termination. The cumulative iteration curve remains close to linear, which is consistent with bounded per-iteration work and the absence of runtime failures. The relatively small threshold-satisfying fraction reflects the strict joint target under continuous scenario rotation, not merely the quality of a fixed localization configuration. Panel (c) shows the parameter drift induced by online adaptation, while panel (d) summarizes the corresponding quality signals and retuning load by hour.

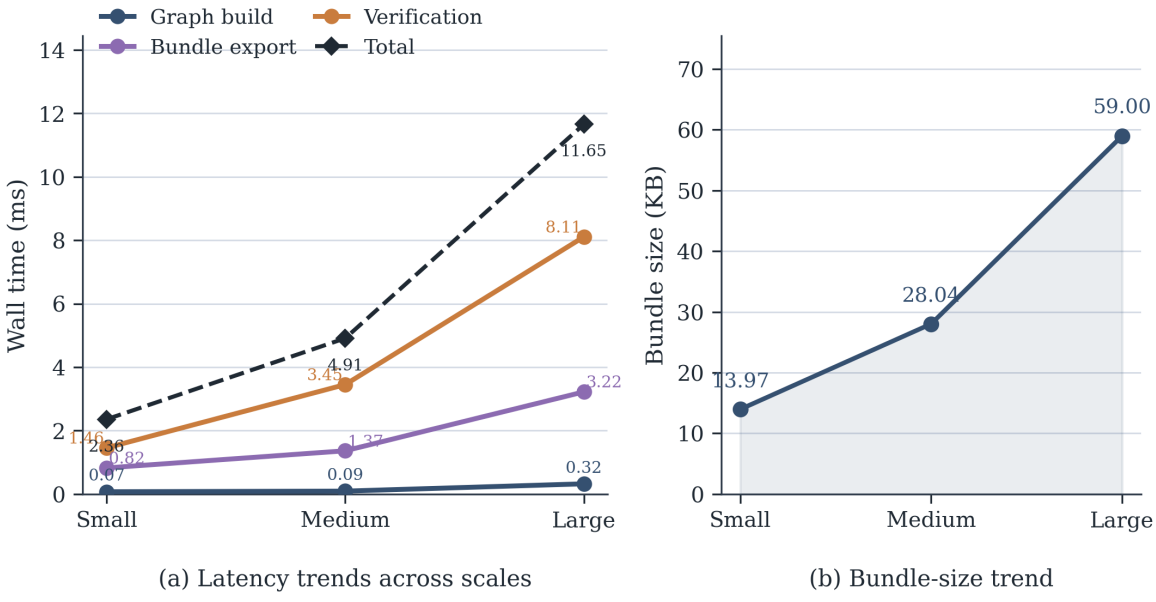


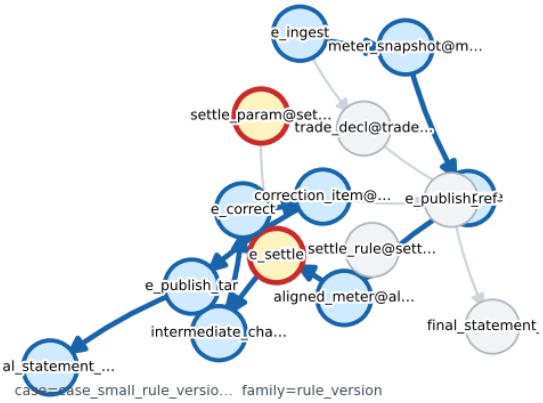
Figure 11. Operational overhead versus scale in the primary study.

 doi:10.20944/preprints202604.1623.v1

The scale-overhead plot (Figure 11) emphasizes trend rather than stacked composition on the same ten-repeat protocol as the primary tables. Graph construction dominates wall time as the template grows across *small*, *medium*, and *large* scales, while bundle export and verification grow more slowly in relative terms; the dashed total line highlights the aggregate growth. Serialized bundle sizes increase with template size but remain modest for offline dispute review on the reported grid. Extrapolation beyond the largest exercised template still requires separate profiling for production-sized graphs.

6.2.5. End-to-End Case Study

Figure 12 presents a single representative replay (small scale, rule-version discrepancy family, linear topology, fixed random seed, Top-10 candidate list when the pool is large enough): (i) the explanation path on the dependency graph with ground-truth highlighting, (ii) the leading ranked candidates, (iii) a shortened evidence-bundle excerpt, and (iv) verifier outcomes on a clean bundle versus a digest-mismatch tamper as defined in Section 6.1.



(a) Impacted path on dependency graph

Rank	Candidate	Score	GT
1	e_ingest	0.768	✗
2	e_publish_tar	0.706	✗
3	correction_item@co...	0.651	✗
4	intermediate_charg...	0.631	✗
5	final_statement_ta...	0.631	✗
6	e_settle	0.626	✓
7	aligned_meter@alig...	0.588	✗
8	e_prepare	0.573	✗
9	meter_snapshot@met...	0.559	✗
10	trade_decl@trade_d...	0.559	✗

(b) Top-10 candidates


```
{{
  "case_id": "case_small_rule_version_42",
  "signed": True,
  "records": [
    { "id": "e_ingest", "type": "event",
      "digest": "ad8a93989edfb30a…", "prev": "null" },
    { "id": "e_prepare", "type": "event",
      "digest": "7828a5940f3f4124…", "prev": "ad8a93989edf…" },
    …
  ],
  "record_count": 15
}}
```


(c) Evidence bundle excerpt

```
Clean bundle
————————————
ok: True
failure_code: –
details: Bundle verification succeeded (V1--V4 checks passe…
Tampered (digest mismatch on last record)
————————————
ok: False
failure_code: E_DIGEST_MISMATCH
details: Digest verification failed.
```

(d) Verification (clean vs tampered)

Figure 12. End-to-end case study.

Taken together, the four panels connect ranked suspects, bundle evidence, and verification outcomes on one deterministic replay. The clean bundle accepts under the configured checks, whereas the digest-mismatch tamper is engineered to fail integrity validation; juxtaposing the two states illustrates how localization outputs are *conditional* on verifier success in the full RootTrace pipeline rather than being presented as free-standing assertions.

7. Discussion and Limitations

7.1. When Trusted Traceability Is Necessary (and When It May Not Be)

RootTrace is motivated by settings in which discrepancy investigations require verifiable evidence across organizational boundaries and in which version drift or repudiation is a realistic concern. When all pipeline stages are operated by a single trusted party and conventional logs are already treated as authoritative, a full evidence-chain design may be unnecessary; lighter-weight reconciliation tooling may suffice. Trusted traceability is therefore best viewed as a targeted response to high-accountability environments rather than as a universal replacement for ordinary observability tooling. It is most

 doi:10.20944/preprints202604.1623.v1

justified when the cost of unresolved disputes, unverifiable version claims, or cross-organization blame assignment exceeds the additional recording and governance overhead introduced by RootTrace.

7.2. Deployment Considerations

Governance and key management. Non-repudiation presupposes explicit rules for identities and keys, who may issue credentials, how trust roots are published, and how relying parties detect mis-issuance or compromise. Public, append-only transparency logs provide a concrete pattern for publishing integrity-relevant metadata so third parties can detect forked or inconsistent views of the same logical log [13]. Key transparency and related commitment-based designs further illustrate how public verifiability can be separated from payload disclosure when interfaces are engineered for selective revelation [16]. Where logs themselves become evidence, recent system-audit work also stresses coverage, availability, and isolation against privileged producers [11,12]. Independent of keying mechanics, several adjacent strands inform how organizations justify and harden evidence-producing stacks. Economic treatments that integrate cost–benefit reasoning with standardized cybersecurity frameworks clarify how control investments can be prioritized under risk [20]. A systematic taxonomy of attacks on open-source software supply chains highlights compromise paths that apply to dependencies of audit and settlement software alike [21]. Runtime causal instrumentation for distributed services helps operators relate symptoms to upstream request paths during investigations [22]. A survey of threats against industrial IoT-oriented operational networks summarizes integrity and availability pressures on field-adjacent data systems [23]. Evidence on cross-country inequality in European electricity prices underscores why comparable, well-documented price inputs matter when parties compare market outcomes [24]. Crisis-period public-sector governance studies emphasize rapid cross-agency information coordination under disruption [25]. A plausible governance model is a designated market-trust authority, or a jointly governed settlement committee, that maintains the actor registry, issues or approves verification keys, publishes trust-root updates, and archives checkpoint references. Data providers and the settlement operator remain responsible for signing their own records, while auditors and dispute-resolution parties consume the published trust material in read-only form. The arrangement is intentionally lightweight but concrete enough to anchor later deployment discussion.

Operational overhead. Evidence recording and verification introduce latency and storage overhead. Section 6 reports measured costs on the default small/medium grid and scale-overhead trends (Figure 11); production feasibility still depends on workloads beyond that benchmark.

System integration. Integrating evidence collection into existing settlement pipelines may require nontrivial engineering effort and organizational coordination. The offline simulator in Section 6 provides a controlled evaluation environment, but it does not capture every operational edge case.

7.3. Limitations

Data availability and representativeness. Conclusions about usefulness and overhead depend on the availability of representative artifacts and discrepancy scenarios. Synthetic data support controlled experiments, but they do not fully capture operational complexity. In particular, the gap relative to operational industrial practice is driven mainly by simulator abstraction and by the idealized, checker-aligned tamper suite of Section 6.1; we therefore scope Section 6 as reproducible stress-testing of the claimed mechanisms rather than as calibrated forecasts of field-side rates or tail risk.

Granularity trade-offs. Localization quality depends on the traceability boundary and the granularity of recorded dependencies. Coarser boundaries may yield large suspect sets; finer boundaries may increase overhead. Sensitivity to these choices can be studied by varying the recording policy; the primary tables use fixed hyperparameters unless noted (e.g., the adaptive endpoint in Table 6).

Threat model scope. RootTrace provides tamper evidence and verifiability within the evidence recording boundary. It does not, by itself, guarantee correctness of the settlement computation beyond what can be audited from recorded inputs, versions, and events. The high verification and tamper-

 doi:10.20944/preprints202604.1623.v1

detection rates in Section 6.2.2 characterize a *reference* fault injection suite aligned with the implemented checks; they should not be read as a security proof against adaptive or distribution-shift attacks.

Verification ceiling. Saturated detection on seven scripted classes does not imply near-zero false negatives on unseen tamper distributions; extending the evaluation would require additional mutation models and, where applicable, signed-bundle baselines for every tamper class.

Privacy constraints. Minimal disclosure via commitments and metadata may limit the ability to fully reproduce computations in an external audit, depending on access policy. The paper therefore treats privacy as a deployment constraint rather than claiming a universal privacy solution.

8. Conclusions

The paper introduced RootTrace for settlement discrepancy localization in multi-party electricity trading and framed the task as evidence-backed path inference rather than as a purely numerical reconciliation problem. RootTrace combines discrepancy-aware dependency modeling, SDPC-based propagation and constraint-guided ranking, and a trusted recording and verification layer that supports auditable claims under limited disclosure. The reported experiments isolate these pieces on a semi-synthetic benchmark (Section 6): mean localization and analyst-effort metrics come from ten full-grid replications with frozen SDPC weights; verification rates characterize a scripted tamper suite aligned with the reference checker; ablations disable single modules while holding the remainder fixed; and the eight-hour stress log exercises online adaptation under stricter per-iteration success rules than the primary tables. Future work will extend the benchmark with additional real settlement extracts, relax the verification ceiling with richer fault models, refine evidence-completeness formalisms, and evaluate deployment behavior at larger operational scale.

Author Contributions: Conceptualization, Pingyan Mo and Kai Li; methodology, Pingyan Mo, Xihong Liang, Xin Hu, and Jinwen Xi; software, Jiajun Liu; validation, Pingyan Mo, Kai Li and Xihong Liang; formal analysis, Xihong Liang; investigation, Kai Li; resources, Pingyan Mo; data curation, Pingyan Mo; writing—original draft preparation, Pingyan Mo, Xin Hu, and Jinwen Xi; writing—review and editing, Kai Li, Xihong Liang, Xin Hu, and Jinwen Xi; visualization, Kai Li; supervision, Jiajun Liu; project administration, Pingyan Mo. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study. Written informed consent has been obtained from the patient(s) to publish this paper.

Data Availability Statement: The original data presented in the study are openly available in UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Electricity+Market.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NEM	National Electricity Market
PJM	PJM Interconnection
PROV	W3C PROV Data Model
SDPC	Settlement Discrepancy Propagation Chain
RCA	Root cause analysis
UDF	User-defined function

References

1. Creutzig, F.; Niamir, L.; Bai, X.; et al. Demand-side solutions to climate change mitigation consistent with high levels of well-being. *Nature Climate Change* **2022**, *12*, 36–46. https://doi.org/10.1038/s41558-021-01219-2.

 doi:10.20944/preprints202604.1623.v1

2. Gielen, D.; Boshell, F.; Saygin, D.; et al. The role of renewable energy in the global energy transformation: A review of potentials and technological issues. *Renewable and Sustainable Energy Reviews* **2019**, *116*, 109414. https://doi.org/10.1016/j.rser.2019.109414.
3. Egerer, J.; Lorenz, C.; Gerbaulet, C. European electricity grid infrastructure expansion in a 2050 context. In Proceedings of the 2013 10th International Conference on the European Energy Market (EEM 2013), Stockholm, Sweden, 2013; pp. 1–7. https://doi.org/10.1109/EEM.2013.6607408.
4. Joskow, P.L. California's electricity crisis. *Oxford Review of Economic Policy* **2001**, *17*, 365–388. https://doi.org/10.1093/oxrep/17.3.365.
5. Marshall, L.; Bruce, A.; MacGill, I. Market mechanisms and technology transition in Australia's National Electricity Market. *Current Sustainable/Renewable Energy Reports* **2022**, *9*, 41–51. https://doi.org/10.1007/s40518-022-00212-3.
6. Walawalkar, R.; Blumsack, S.; Apt, J.; et al. An economic welfare analysis of demand response in the PJM electricity market. *Energy Policy* **2008**, *36*, 3692–3702. https://doi.org/10.1016/j.enpol.2008.06.007.
7. Psallidas, F.; Agrawal, A.; Sugunan, C.; et al. OneProvenance: Efficient extraction of dynamic coarse-grained provenance from database query event logs. *Proceedings of the VLDB Endowment* **2023**, *16*, 3662–3675. https://doi.org/10.14778/3611540.3611555.
8. Yamada, M.; Kitagawa, H.; Amagasa, T.; Matono, A. Augmented lineage: Traceability of data analysis including complex UDF processing. *The VLDB Journal* **2023**, *32*, 963–983. https://doi.org/10.1007/s00778-022-00769-7.
9. Jacques-Silva, G.; Kalyvianaki, E.; Cohn-Gordon, K.; Meguid, A.; Nguyen, H.; Ben-David, D.; Nayak, C.; Saravagi, V.; Stasa, G.; Papagiannis, I.; et al. Unified lineage system: Tracking data provenance at scale. In Proceedings of the Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, 2025; pp. 457–470. https://doi.org/10.1145/3722212.3724458.
10. Datta, P.; Polinsky, I.; Inam, M.A.; et al. ALASTOR: Reconstructing the provenance of serverless intrusions. In Proceedings of the Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 2022; pp. 2443–2460. https://doi.org/10.5555/3691992.3692029.
11. Gandhi, V.; Banerjee, S.; Agrawal, A.; et al. Rethinking system audit architectures for high event coverage and synchronous log availability. In Proceedings of the Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 2023; pp. 391–408. https://doi.org/10.5555/3620237.3620260.
12. Jiang, P.; Huang, R.; Li, D.; et al. Auditing frameworks need resource isolation: A systematic study on the super producer threat to system auditing and its mitigation. In Proceedings of the Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 2023; pp. 355–372. https://doi.org/10.5555/3620237.3620258.
13. Laurie, B. Certificate Transparency. *Communications of the ACM* **2014**, *57*, 40–46. Practice overview; formal protocol in RFC 9162 / RFC 6962., https://doi.org/10.1145/2659897.
14. Ren, X.; Wang, S.; Jin, Z.; et al. Relational debugging: Pinpointing root causes of performance problems. In Proceedings of the Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23), Boston, MA, USA, 2023; pp. 65–80. https://doi.org/10.5555/3605638.3605644.
15. Missier, P.; Belhajjame, K.; Cheney, J. The W3C PROV family of specifications for modelling provenance metadata. In Proceedings of the Proceedings of the 16th International Conference on Extending Database Technology (EDBT 2013), Genoa, Italy, 2013; pp. 773–776. https://doi.org/10.1145/2452376.2452478.
16. Malvai, H.; Kokoris-Kogias, L.; Sonnino, A.; et al. Parakeet: Practical key transparency for end-to-end encrypted messaging. In Proceedings of the Proceedings of the Network and Distributed System Security Symposium (NDSS 2023), The Internet Society, San Diego, CA, USA, 2023. https://doi.org/10.14722/ndss.2023.24545.
17. Nepal, R.; Jamasb, T. Interconnections and market integration in the Irish Single Electricity Market. *Energy Policy* **2012**, *51*, 425–434. https://doi.org/10.1016/j.enpol.2012.08.064.
18. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 2017. OpenReview: https://openreview.net/forum?id=SJU4ayYgl, https://doi.org/10.48550/arXiv.1609.02907.
19. McCracken, M.W.; Ng, S. FRED-MD: A monthly database for macroeconomic research. *Journal of Business and Economic Statistics* **2016**, *34*, 574–589. https://doi.org/10.1080/07350015.2015.1086654.
20. Gordon, L.A.; Loeb, M.P.; Zhou, L. Integrating cost-benefit analysis into the NIST Cybersecurity Framework via the Gordon–Loeb model. *Journal of Cybersecurity* **2020**, *6*, tyaa005. https://doi.org/10.1093/jcyber/tyaa005.

 doi:10.20944/preprints202604.1623.v1

21. Ladisa, P.; Plate, H.; Martinez, M.; et al. SoK: Taxonomy of attacks on open-source software supply chains. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2023; pp. 1509–1526. https://doi.org/10.1109/SP46215.2023.10179304.
22. Mace, J.; Roelke, R.; Fonseca, R. Pivot tracing: Diagnosing causality in distributed systems. In Proceedings of the Proceedings of the 25th Symposium on Operating Systems Principles (SOSP 15), Monterey, CA, USA, 2015; pp. 378–393. https://doi.org/10.1145/2815400.2815415.
23. Al-Hawawreh, M.; Sitnikova, E. Detecting cyber threats in industrial internet of things: A survey. *IEEE Communications Surveys and Tutorials* **2022**, *24*, 2125–2158. https://doi.org/10.1109/COMST.2022.3169443.
24. Bunn, D.W.; Zachmann, G. Inequality in electricity prices: Some evidence from the EU. *Energy Economics* **2012**, *34*, 1372–1379. https://doi.org/10.1016/j.eneco.2012.06.024.
25. Janssen, M.; van der Voort, H. Agile and adaptive governance in crisis response: Lessons from the COVID-19 pandemic. *International Journal of Information Management* **2020**, *55*, 102180. https://doi.org/10.1016/j.ijinfomgt.2020.102180.