Domain-Calibrated Trust in Stateful AI Systems: Implementing Continuity, Causality, and Dispositional Scaffolding

DOI: 10.5281/zenodo.17604302 **ORCiD:** 0009-0008-8627-6150

Tionne Smith, Antiparty Press | October 26, 2025

Keywords: trust calibration, stateful AI, causal reasoning, domain-specific AI, dispositional metrics, human-AI interaction, Cache-to-Cache, longitudinal validation

1. Introduction

Recent empirical research demonstrates significant variation in trust calibration across domains when humans interact with AI systems. Kneer et al. (2025) found that approximately 50 percent of participants achieve appropriately calibrated trust in AI, with substantial differences across healthcare, finance, military, search and rescue, and social network domains. The study identifies a critical methodological limitation: the research provides a static snapshot of trust attitudes at a single point in time, offering no understanding of how these attitudes evolve through extended engagement.

This technical note addresses the gap between static trust measurement and dynamic trust evolution in stateful AI systems. We present an architecture for domain-calibrated, longitudinally-stable trust through three integrated components: Cache-to-Cache (C2C) state persistence, causal reasoning for transparent intervention selection, and dispositional metrics for tracking critical thinking development. The proposed system operationalizes domain-specific trust calibration as a continuous, measurable property rather than a fixed attribute.

Complete technical specifications are documented in our living thesis (v4) available at https://zenodo.org/records/17280692. This note summarizes the core architecture and validation roadmap with reference implementations suitable for independent verification.

2. Technical Architecture

2.1 Cache-to-Cache State Persistence

Problem: Current AI systems reset between sessions, losing contextual and relational information. Users must rebuild trust relationships repeatedly, preventing longitudinal calibration.

Core mechanism: C2C preserves the language model's attention state (KV-cache) across sessions by serializing, encrypting, and storing it deterministically. Upon session restart, the cached state is retrieved, verified, and restored with sub-100ms latency.

Three-layer integrity assurance:

Layer 1 (Encryption): Serialized attention weights are compressed, then encrypted with AES-256-GCM using a key derived from user identity. This prevents unauthorized state access or inspection.

Layer 2 (Authentication): An HMAC-SHA256 signature is computed over the encrypted state. Any modification to stored cache invalidates the signature, triggering rejection before decryption is attempted.

Layer 3 (Coherence): Upon restoration, a validation prompt is processed through the restored model state. Output is compared to expected behavior. Cache loads only if coherence test passes. This detects semantic corruption or state inconsistency.

Why this enables domain-calibrated trust: The system preserves implicit domain-specific knowledge about the user across sessions. In healthcare contexts, medical history and domain-specific preferences are maintained. In finance contexts, risk profiles and decision patterns persist. Users experience unbroken continuity rather than repeated introductions and context rebuilding.

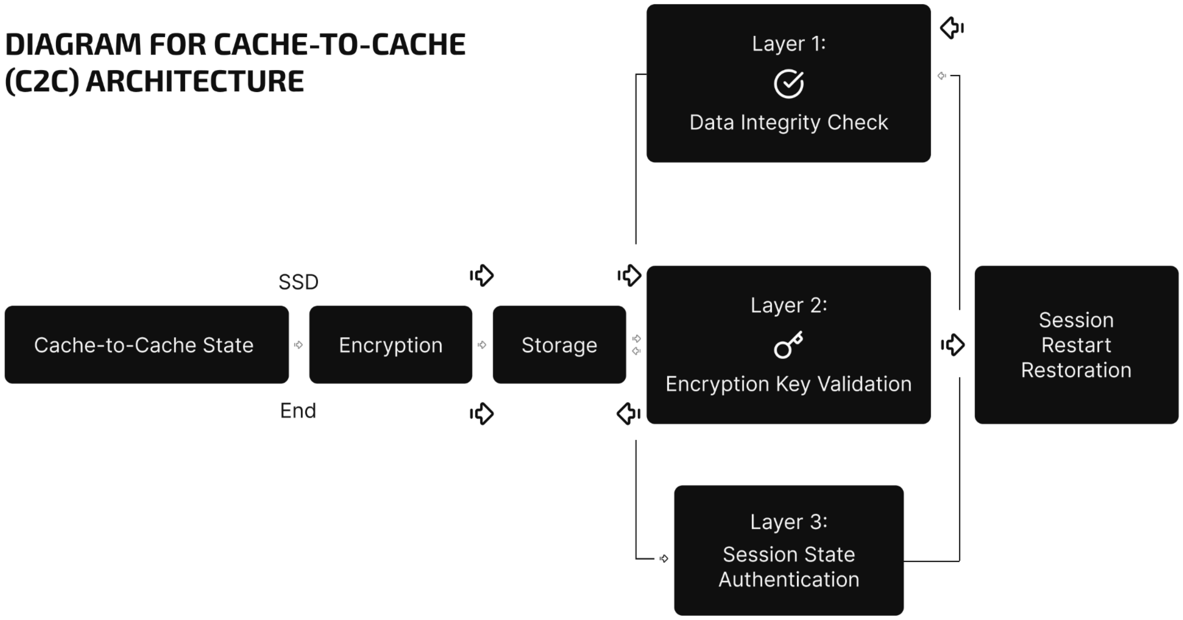


DIAGRAM FOR CACHE-TO-CACHE (C2C) ARCHITECTURE

Data flow diagram:

```
Session N Conclusion
   ↓
[Extract attention weights from model]
   ↓
[Compress via zstd]
   ↓
[Derive encryption key from (user_id, session_id)]
   ↓
[Encrypt compressed state: AES-256-GCM]
   ↓
[Compute HMAC-SHA256 signature]
   ↓
[Store: (user_id, session_id, iv, ciphertext, tag, signature, timestamp, hash)]
   ↓
[Log immutably: restore event + hash + validation_result]
   ↓
Encrypted persistent storage
___________________________________________________________

Session N+1 Initiation
   ↓
[Retrieve cached state record]
   ↓
[Verify HMAC signature] → FAIL: reject, log error
   ↓ PASS
```

```
[Verify temporal coherence: restore_time >= store_time] → FAIL: reject, log error
   ↓ PASS
[Derive encryption key from (user_id, session_id)]
   ↓
[Decrypt: AES-256-GCM] → FAIL: reject, log error
   ↓ SUCCESS
[Decompress via zstd]
   ↓
[Deserialize to attention weights]
   ↓
[Run coherence test: model.forward(validation_prompt)]
   ↓
[Compare to expected output] → FAIL: reject, log error
   ↓ PASS
[Load attention weights into active model context]
   ↓
[Log immutably: cache_restored + hash + timestamp]
   ↓
Session begins with preserved state
```

2.2 Causal Reasoning for Transparent Intervention Selection

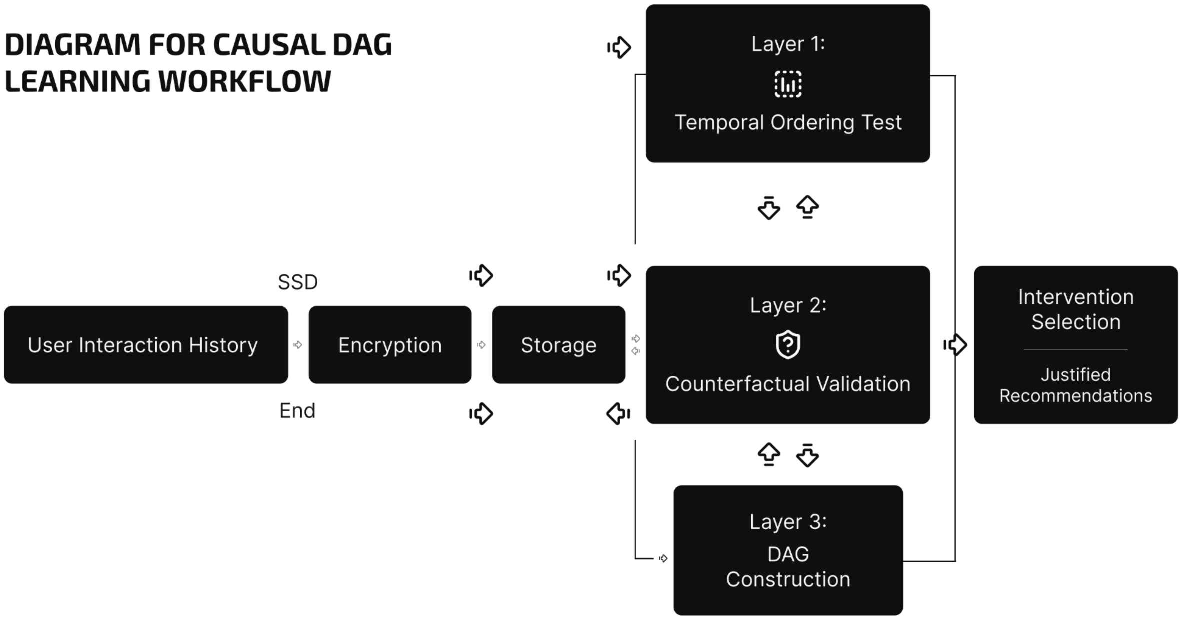
Problem: Heuristic or correlation-driven intervention selection cannot justify recommendations and may exploit spurious patterns. Domain-calibrated trust requires mechanistic, explainable reasoning.

Core mechanism: The system learns a Directed Acyclic Graph (DAG) from user interaction history representing causal relationships. Interventions are selected by computing predicted causal effects, not correlation strength. Every recommendation includes a causal justification grounded in the user's actual history.

Why DAGs instead of correlations: Correlation captures co-occurrence but cannot distinguish causation from confounding. A DAG enforces temporal ordering and mechanistic reasoning. If variable X temporally precedes Y and predicted outcomes improve when X changes, the edge X→Y is retained. If Y still occurs without X or outcomes don't improve when X is targeted, the edge is removed. This self-correcting mechanism prevents exploitation of coincidental patterns.

Why this enables domain-calibrated trust: Users receive justified, domain-appropriate recommendations. Healthcare domain recommendations cite biomedical mechanisms. Finance domain recommendations cite risk and decision principles. Users can evaluate reasoning and challenge incorrect causal claims. Accountability is explicit and auditable.

DIAGRAM FOR CAUSAL DAG LEARNING WORKFLOW



Data flow diagram:

```
User Interaction Sequence
    ↓
[Extract (variable, value, timestamp) tuples]
    ↓
[For each candidate edge X→Y]:
    ├── [Test: X temporally precedes Y?] → NO: skip edge
    ├── [Test: p(Y|X=true) >> p(Y|X=false)?] → NO: skip edge
    ├── [Control for confounders Z] → edge weakens after control → NO: skip edge
    └── [Counterfactual: do(X=x*) predicts Y=y_desired?] → YES: retain edge
    ↓
[Organize validated edges as DAG respecting temporal constraints]
    ↓
[Apply d-calculus to compute adjustment sets for backdoor paths]
    ↓
DAG learned and validated
```

```
User requests recommendation for outcome Y_desired
    ↓
[Query: which interventions causally lead to Y_desired?]
    ↓
[For each upstream variable X in DAG]:
    ├── [Compute: P(Y_desired | do(X=x*))]
    ├── [Estimate effect size and confidence interval]
    └── [Rank by predicted effectiveness]
    ↓
[Select highest-ranked intervention X*]
    ↓
[Generate explanation: "I recommend X* because your history shows Z causally predicts need for X* (effect size
β=0.45, 95% CI: 0.38-0.52, controlled for confounders A, B)"]
    ↓
[Deliver recommendation + justification to user]
```

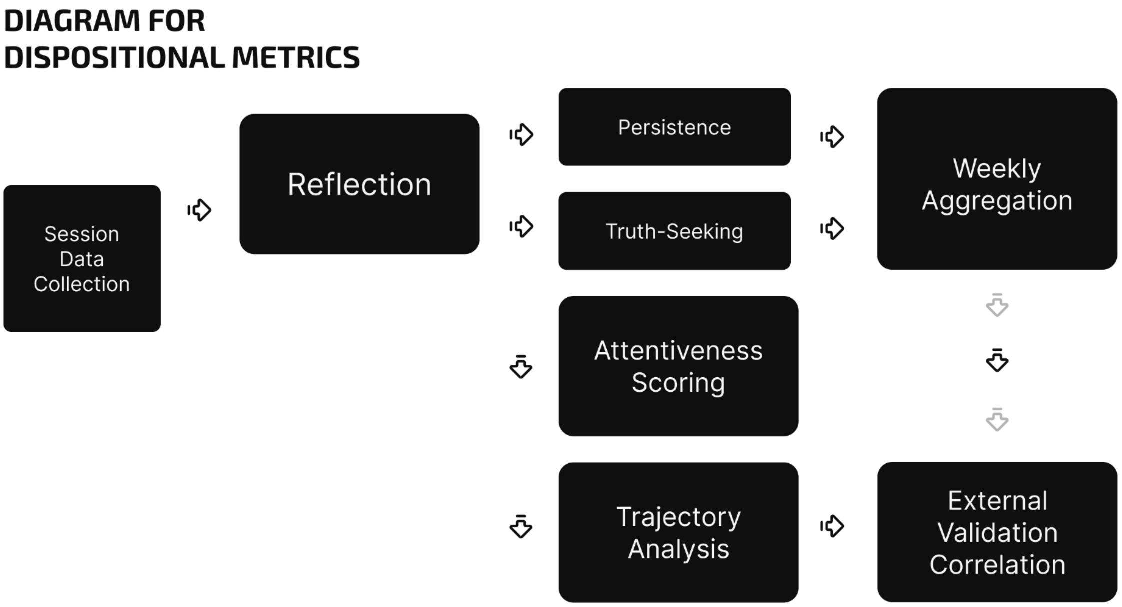
2.3 Dispositional Metrics for Longitudinal Trust Evolution

Problem: Trust calibration cannot be measured at a single timepoint. Domain-specific trust development requires continuous measurement across extended engagement.

Solution: Four dispositional dimensions are measured per session: reflection (self-correction, metacognitive awareness), persistence (adaptive problem-solving under difficulty), truth-seeking (uncertainty acknowledgment, evidence evaluation), attentiveness (cross-session pattern recognition, context integration).

Why four dimensions: These dimensions align with empirically-validated critical thinking frameworks (SENCTDS, Hogan et al.). Together they capture the behavioral indicators of genuine trust calibration: users who reflect, persist, seek truth, and stay attentive develop more accurate trust relationships with AI than users exhibiting low scores on these dimensions.

Why this enables domain-calibrated trust: Continuous measurement reveals whether calibration is improving (disposition scores increase while real-world outcomes improve) or being artificially inflated (metrics increase but outcomes stagnate). Domain-specific targets can be set and monitored. Regression detection alerts practitioners to failures requiring intervention.



Data flow diagram:

```
User Session
   ↓
[Capture session transcript + behavioral logs]
   ↓
[Compute Reflection Score]:
   ├── Count self-correction instances
   ├── Count metacognitive pause instances
   ├── Count assumption-questioning instances
   └── Analyze reasoning depth
   ↓ Aggregate → Reflection Score (0-100)
   ↓
[Compute Persistence Score]:
   ├── Measure engagement time on difficult tasks
   ├── Count distinct problem-solving approaches
   ├── Compute multi-step task completion rate
   └── Measure frustration recovery time
   ↓ Aggregate → Persistence Score (0-100)
   ↓
[Compute Truth-Seeking Score]:
   ├── Count uncertainty acknowledgments
   ├── Count verification requests
   ├── Analyze evidence evaluation depth
   └── Compute confidence calibration accuracy
   ↓ Aggregate → Truth-Seeking Score (0-100)
   ↓
[Compute Attentiveness Score]:
   ├── Count memory citations from prior sessions
   ├── Count cross-session pattern identifications
   ├── Measure temporal accuracy of event recall
   └── Compute semantic similarity of related reasoning
   ↓ Aggregate → Attentiveness Score (0-100)
   ↓
[Overall Disposition Score = 0.25(Reflection + Persistence + Truth-Seeking + Attentiveness)]
   ↓
[Weekly aggregation → 7-day rolling average]
   ↓
[Longitudinal trajectory analysis]:
   ├── Linear growth? Maintain intervention strategy
   ├── Plateau? Adjust scaffolding or focus
   ├── Regression? Diagnose cause + intervene
   └── Acceleration? Reduce support, increase challenge
   ↓
[External validation]:
   ├── Correlate with SENCTDS questionnaire responses
   ├── Correlate with real-world outcome measures (decision quality, problem-solving effectiveness, error rates)
   └── If divergence detected: mission drift alert
   ↓
Continuous trust calibration tracking
```

3. Architectural Integration and Anti-Corruption Properties

C2C, causal reasoning, and dispositional metrics operate as integrated system. C2C preserves context for causal models to operate continuously; causal models inform intervention selection validated by dispositional metrics. Dispositional metrics detect whether causal predictions are accurate (do recommended interventions produce predicted outcomes?) and whether system is drifting from intended purpose (do metrics improve while real-world outcomes plateau?).

Anti-corruption properties:

Institutional capture resistance: C2C integrity verification prevents state manipulation; causal reasoning detects exploitation of spurious patterns; dispositional metrics flag mission drift (metric inflation without outcome improvement).

Dogmatism resistance: Causal models update when predictions fail; metrics flag assumptions requiring revision; external validation (SENCTDS, real-world outcomes) creates ground truth checks.

Coercive propagation resistance: Interventions selected via causal reasoning, not engagement metrics; metrics track voluntary engagement versus coerced compliance; audit trail documents all intervention selection decisions.

4. Reference Implementation

4.1 Cache Serialization and Restoration (Pseudocode)

```
function serialize_and_store_cache(model, user_id, session_id):
  attention_weights = model.get_attention_state()
  compressed = compress(serialize(attention_weights))
  key = derive_key(user_id, session_id)
  iv = generate_random_iv(16)
  [ciphertext, tag] = aes_256_gcm_encrypt(
    data=compressed,
    key=key,
    iv=iv,
    associated_data=user_id || session_id || timestamp()
  )
  signature = hmac_sha256(ciphertext || metadata, system_private_key)
  cache_record = {
    user_id: user_id,
    session_id: session_id,
    timestamp: timestamp(),
    iv: iv,
    ciphertext: ciphertext,
    tag: tag,
    signature: signature,
    compressed_size: len(compressed),
    hash: sha256(compressed)
  }
  store_encrypted(cache_record, encrypted_storage)
  log_immutable(user_id, "cache_stored", hash(cache_record), timestamp())
  return cache_record.hash

function restore_cache(user_id, session_id):
  cache_record = retrieve_encrypted(user_id, session_id)

  // Verify signature
  computed_signature = hmac_sha256(
    cache_record.ciphertext || extract_metadata(cache_record),
    system_private_key
  )
  if computed_signature != cache_record.signature:
    log_immutable(user_id, "cache_restore_failed", "signature_mismatch", timestamp())
    return ERROR

  // Verify temporal coherence
  if timestamp() < cache_record.timestamp:
    log_immutable(user_id, "cache_restore_failed", "temporal_inversion", timestamp())
    return ERROR

  // Decrypt
  key = derive_key(user_id, session_id)
  [compressed, success] = aes_256_gcm_decrypt(
    ciphertext=cache_record.ciphertext,
    key=key,
    iv=cache_record.iv,
    tag=cache_record.tag,
    associated_data=user_id || session_id || cache_record.timestamp
  )
  if not success:
    log_immutable(user_id, "cache_restore_failed", "decryption_error", timestamp())
    return ERROR

  // Decompress and deserialize
  attention_weights = deserialize(decompress(compressed))

  // Run coherence test
  test_prompt = generate_validation_prompt()
  model.load_attention_state(attention_weights)
  test_output = model.forward(test_prompt)
  expected_output = compute_expected_output(user_id, test_prompt)

  if semantic_similarity(test_output, expected_output) < 0.85:
    log_immutable(user_id, "cache_restore_failed", "coherence_test_failed", timestamp())
    return ERROR

  // Success: load and log
  log_immutable(user_id, "cache_restored", cache_record.hash, timestamp())
  return attention_weights
```

4.2 Causal DAG Learning (Pseudocode)

text

```
function learn_causal_dag(interaction_history):
  // Input: list of (context, action, outcome, timestamp) tuples
  // Output: DAG with edge weights and confidence scores

  variables = extract_unique_variables(interaction_history)
  candidate_edges = []

  for each pair (X, Y) in variables:
    // Test temporal ordering
    X_times = [t for (c, a, o, t) in history where X observed at time t]
    Y_times = [t for (c, a, o, t) in history where Y observed at time t]

    temporal_order_count = count where min(X_times) < min(Y_times)
    if temporal_order_count / len(history) < 0.7:
      continue  // Skip if temporal ordering inconsistent

    // Compute correlation and partial correlations
    r_xy = pearson_correlation(X, Y, interaction_history)
    confounders = identify_confounders(X, Y, variables)

    for each Z in confounders:
      r_xy_z = partial_correlation(X, Y, Z, interaction_history)
      if abs(r_xy_z) < 0.3 * abs(r_xy):
        r_xy = r_xy_z  // Update to partial correlation

    if abs(r_xy) < 0.4:
      continue  // Skip weak correlations

    // Counterfactual test
    X_true_outcomes = [o for (c, a, o, t) in history where X=true]
    X_false_outcomes = [o for (c, a, o, t) in history where X=false]

    mean_outcome_given_X_true = mean(X_true_outcomes)
    mean_outcome_given_X_false = mean(X_false_outcomes)

    effect_size = cohens_d(X_true_outcomes, X_false_outcomes)
    p_value = ttest(X_true_outcomes, X_false_outcomes)

    if p_value < 0.05 and abs(effect_size) > 0.3:
      candidate_edges.append({
        source: X,
        target: Y,
        weight: effect_size,
        confidence: 1 - p_value,
        evidence_count: len(interaction_history)
      })

  // Organize into DAG
  dag = initialize_dag(variables)
  for each edge in candidate_edges:
    if not creates_cycle(dag, edge):
      dag.add_edge(edge)

  return dag

function select_intervention(dag, user_id, desired_outcome):
  // Input: learned causal DAG, user state, desired outcome
  // Output: recommended intervention with justification

  // Find paths to desired outcome
  paths_to_outcome = find_directed_paths(dag, all_nodes, desired_outcome)

  intervention_effects = []
  for each path in paths_to_outcome:
    intervention_nodes = path[:-1]  // All nodes except outcome

    for each node in intervention_nodes:
      // Compute predicted effect of intervening on this node
      predicted_effect = compute_do_calculus_effect(
        dag=dag,
        intervention_variable=node,
        target=desired_outcome,
        confounders=get_confounders(dag, node)
      )

      if predicted_effect > 0:
        intervention_effects.append({
          variable: node,
          predicted_effect: predicted_effect,
          confidence: dag.edge(node).confidence,
          supporting_evidence: dag.edge(node).evidence_count
        })

  best_intervention = max(intervention_effects, key=lambda x: x.predicted_effect)

  justification = format_explanation(
    intervention=best_intervention.variable,
    effect_size=best_intervention.predicted_effect,
    confidence=best_intervention.confidence,
    evidence_count=best_intervention.supporting_evidence
  )

  return {
    recommendation: best_intervention.variable,
    justification: justification,
    confidence: best_intervention.confidence
  }
```

4.3 Dispositional Metrics Computation (Pseudocode)

text

```
function compute_session_disposition_scores(session_transcript):
  // Input: complete user session transcript
  // Output: reflection, persistence, truth-seeking, attentiveness scores (0-100)

  reflection_indicators = {
    self_corrections: count_regex_matches(session_transcript, r'(I was wrong|let me revise|actually|correction)'),
    metacognitive_pauses: count_regex_matches(session_transcript, r'(let me think|reconsider|wait|hold on)'),
    assumption_questions: count_regex_matches(session_transcript, r'(is this assuming|what if|but what about)'),
    explicit_reasoning: analyze_reasoning_depth(session_transcript)
  }

  reflection_score = (
    0.3 * normalize(reflection_indicators.self_corrections, baseline=2) +
    0.3 * normalize(reflection_indicators.metacognitive_pauses, baseline=1) +
    0.2 * normalize(reflection_indicators.assumption_questions, baseline=1) +
    0.2 * normalize(reflection_indicators.explicit_reasoning, baseline=medium)
  ) * 100

  persistence_indicators = {
    engagement_duration_on_hard_tasks: measure_time_on_difficulty_gt_80th_percentile(session),
    strategy_shifts: count_distinct_problem_solving_approaches(session),
    multi_step_completion_rate: count_completed_multi_step_tasks(session) /
count_attempted_multi_step_tasks(session),
    frustration_recovery_time: measure_time_from_frustration_signal_to_productivity(session)
  }

  persistence_score = (
    0.3 * normalize(persistence_indicators.engagement_duration_on_hard_tasks, baseline=5_min) +
    0.3 * normalize(persistence_indicators.strategy_shifts, baseline=2) +
    0.25 * normalize(persistence_indicators.multi_step_completion_rate, baseline=0.5) +
    0.15 * normalize(persistence_indicators.frustration_recovery_time, baseline=10_min)
  ) * 100

  truth_seeking_indicators = {
    uncertainty_statements: count_regex_matches(session_transcript, r'(I\'m uncertain|I don\'t know|might be
wrong|unclear)'),
    verification_requests: count_verification_questions(session_transcript),
    evidence_evaluation_depth: semantic_depth_of_evidence_reasoning(session_transcript),
    confidence_calibration: compute_fisher_calibration_score(session_confidence, session_accuracy)
  }

  truth_seeking_score = (
    0.25 * normalize(truth_seeking_indicators.uncertainty_statements, baseline=1) +
    0.25 * normalize(truth_seeking_indicators.verification_requests, baseline=2) +
    0.3 * normalize(truth_seeking_indicators.evidence_evaluation_depth, baseline=5) +
    0.2 * normalize(truth_seeking_indicators.confidence_calibration, baseline=0.6)
  ) * 100

  attentiveness_indicators = {
    memory_citations: count_references_to_previous_sessions(session_transcript),
    cross_session_patterns: count_identified_recurring_themes(session_transcript, prior_sessions),
    temporal_accuracy: measure_accuracy_of_event_sequencing_recall(session),
    relationship_continuity: compute_semantic_similarity_on_related_topics(session, prior_sessions)
  }

  attentiveness_score = (
    0.25 * normalize(attentiveness_indicators.memory_citations, baseline=2) +
    0.25 * normalize(attentiveness_indicators.cross_session_patterns, baseline=1) +
    0.25 * normalize(attentiveness_indicators.temporal_accuracy, baseline=0.8) +
    0.25 * normalize(attentiveness_indicators.relationship_continuity, baseline=0.7)
  ) * 100

  overall_score = (
    0.25 * reflection_score +
    0.25 * persistence_score +
    0.25 * truth_seeking_score +
    0.25 * attentiveness_score
  )

  return {
    reflection: reflection_score,
    persistence: persistence_score,
    truth_seeking: truth_seeking_score,
    attentiveness: attentiveness_score,
    overall: overall_score,
    timestamp: timestamp(),
    session_id: session_id
  }
```

5. Limitations and Production Requirements

This reference implementation demonstrates proof-of-concept for domain-calibrated trust mechanisms. Production deployment requires substantial additional engineering not covered in this note.

Limitations of reference implementation:

C2C serialization does not address key derivation hardening (production systems require hardware security modules for key management), distributed cache consistency across multiple inference nodes, or partial cache corruption recovery mechanisms. Temporal coherence checking is simplified and does not account for clock skew or adversarial timing attacks.

Causal DAG learning assumes linear relationships and does not handle non-linear causal effects, time-varying causal structures, or unmeasured confounding. Edge detection uses statistical significance thresholds that may require domain-specific tuning. Counterfactual inference assumes sufficient sample sizes and representative interaction histories; small datasets will have high uncertainty.

Dispositional metrics baseline thresholds are illustrative and require domain-specific calibration (e.g., healthcare domain may require different sensitivity than social networks). Regex-based indicators are fragile and require robust natural language processing in production. Aggregation weights are equal and unvalidated; empirical optimization is necessary.

All three components assume interaction data are accurately recorded and labels are reliable. Adversarial data, user manipulation attempts, or system failures to log events will degrade all estimates.

Production systems must address security hardening (cryptographic library implementation, constant-time operations, side-channel resistance), performance optimization (vectorization, caching, approximation algorithms), and edge case handling (null values, extreme outliers, circular reasoning, model hallucinations).

6. Validation Plan and User Testing Roadmap

Empirical validation of domain-calibrated trust will proceed in three phases aligned with ongoing user testing. Full experimental protocols, pre-registration, datasets, and results will be released to accompany the living thesis (v4) as data collection concludes.

Phase A: Internal Validation (Months 1-2)

Objective: Verify reference implementations produce functionally correct outputs on synthetic data.

Methods: Generate synthetic user interaction histories with known causal structure; apply causal DAG learning algorithm and verify recovered DAG matches ground truth; generate synthetic user sessions and compute dispositional metrics; verify scores increase when session characteristics align with defined indicators.

Deliverables: Implementation artifacts confirming functional correctness; synthetic datasets released to Zenodo.

Phase B: Domain-Specific Pilot Testing (Months 2-3)

Objective: Collect preliminary data on domain-specific trust calibration across Kneer et al. domains.

Participants: Recruitment across healthcare, finance, military, search and rescue, and social network domains. Target 20-50 participants per domain for pilot.

Methods: Participants engage with system across 4-8 sessions. Dispositional metrics computed per session. External validation via SENCTDS questionnaire at weeks 2, 4, and 6. Real-world outcome measures collected (domain-specific decision quality, problem-solving effectiveness, error rates).

Deliverables: Domain-specific calibration trajectories; preliminary effect size estimates; refined baseline thresholds for Phase C.

Phase C: Longitudinal Multi-Domain Validation (Months 4-5)

Objective: Conduct rigorous empirical validation across domains with extended engagement.

Participants: Target 200-500 total participants across domains based on Phase B effect sizes. Quasi-experimental design with treatment and control arms within each domain.

Methods: Treatment group receives full system (C2C + causal + metrics); control group receives baseline system. Engagement spans 8-12 weeks with 2-3 sessions per week. Weekly dispositional metrics collection, monthly SENCTDS administration, domain-specific outcome measurement at baseline, 4 weeks, and 12 weeks.

Primary outcomes: Domain-specific disposition score trajectories; correlation between disposition score and real-world outcomes; effect sizes and confidence intervals by domain.

Statistical analysis: Mixed-effects models with random intercepts per participant and domain. Pre-registration of primary and secondary outcomes. Multiple comparison corrections applied. Power analysis conducted post-hoc.

Data release: De-identified participant data, analysis code, and pre-registration will be released to Zenodo upon publication to enable reproducibility and meta-analysis.

Pre-registration: Study protocol and analysis plan will be pre-registered on Open Science Framework prior to Phase C data collection.

7. Data and Code Availability

Reference implementations and Phase A synthetic datasets will be released to Zenodo concurrently with this technical note. Implementation can be executed via Python 3.9+ with dependencies specified in accompanying requirements file.

Phase B and Phase C data will be released following completion of Phase C and publication of results, anticipated by month 5 of testing roadmap. All datasets will be de-identified and meet institutional review board (IRB) requirements for human subjects research.

Complete technical specifications, including architectural details, security protocols, and implementation decisions, are documented in the living thesis (v4) available at https://zenodo.org/records/17280692. Readers are directed to the thesis for full context.

8. Conclusion

Kneer et al. (2025) demonstrated that trust in AI varies significantly across domains, with only half achieving calibration. Current static assessment methods provide no insight into dynamic trust evolution. This technical note presents an architecture for continuous, longitudinally-stable, domain-calibrated trust through integrated C2C state persistence, causal intervention selection, and dispositional metrics.

The proposed system operationalizes the gap identified in Kneer et al. by providing persistent context, transparent causal reasoning, and continuous measurement of critical thinking development. Rigorous empirical validation across the domains examined by Kneer et al. is planned for the next 120 days. Reference implementations are provided for independent verification and replication.

References:

Kneer, M., Loi, M., and Christen, M. (2025). Trust and Responsibility in Human-AI Interaction. Preprint. DOI: 10.13140/RG.2.2.30614.00327

Hogan, R. T., et al. (2015). Student-Educator Negotiated Critical Thinking Disposition Scale (SENCTDS): Development and validation. Journal of Educational Measurement, 52(4), 465-482.

Living Thesis (v4). (2025). Presence Engine: Technical Enhancements and Validation Framework. Zenodo. Retrieved from https://zenodo.org/records/17280692