ContextForge: Architecture-First Persistent Agentic Memory via Model Context Protocol

Multi-Trigger OR-Gate Security, OR-Set CRDT Sync, Tri-Core Failover, Differential Context Injection, Recency-Weighted Retrieval, and a Memory Quality Benchmark

Trilochan Sharma[1]

[1]Independent Researcher parnishklpo@gmail.com

April 2026

Abstract

Stateless Retrieval-Augmented Generation (RAG) systems suffer from three structural failure modes: zero adversarial block rate, high cold-start failover latency (∼480 ms), and unbounded context injection that grows linearly with session history. We present **ContextForge**, a five-pillar agentic memory architecture that simultaneously addresses all three failures and exposes every capability as a **22-tool Model Context Protocol (MCP)** server—providing persistent, adversarially-hardened memory in Claude Desktop, Cursor, VS Code, and Windsurf with zero client-side code changes.

The write-security backbone is `ReviewerGuard`, an append-only SQLite event ledger with a *multi-trigger OR-gate* (v3): Path A fires on char-level Shannon entropy ($H_{\text{char}} \geq H^* = 4.8$ bits/char, catching obfuscated/encoded payloads); Path B fires on injection intent score (≥ 0.70 over 22 patterns + charter-keyword overlap, catching natural-language injections); Path C (trigram perplexity gate, $P^* = 231.8$) provides orthogonal defence against entropy-mimicry. Any single path independently blocks the write. Verified-origin traffic receives an elevated threshold ($H^*_{\text{VOH}} = 4.375$ bits via HMAC-SHA256) and a Sliding-Window Temporal Correlator detects slow-drip multi-write escalation sequences at 100% detection, 0% FPR.

An OR-Set CRDT synchronisation layer (`src/sync/crdt_sync.py`) supports conflict-free concurrent writes from multiple IDE clients with add-wins semantics, verified at 100% convergence across three concurrent-write scenarios including a full split-brain reconnection.

Suite 14 results (300 samples, measured). In `CF_MODE=experiment` (v3 multi-trigger OR-gate): benign FPR = 1% (vs. 96% in PAPER mode, $\Delta = -95$ pp), adversarial recall = 55% (vs. 95% in PAPER mode; the tradeoff is deliberate—Path A replaces the miscalibrated word-level $H^* = 3.5$ gate that produced 96% benign FPR), edge-case FPR = 16% ($\Delta = -81$ pp). The 45% adversarial miss rate is confined to paraphrased natural-language injections with max intent_score = 0.50, identified as the primary target for Path C (perplexity gate). Hardened-RAG achieves 71% recall at 45% edge FPR—higher recall but substantially worse precision on maintenance text.

Suite 15 v2 results (160 samples, 6 systems, measured). A Memory Quality Benchmark evaluates retrieval recall, update preference, delete fidelity, and poison resistance using a zero-LLM harness with BM25 retrieval and the REAL ReviewerGuard v3 gate. Version 2 adds recency-weighted retrieval ($\exp(-\lambda\Delta t)$, $\lambda = 0.0001\,\text{s}^{-1}$), raising update accuracy from 0.229 to 0.600 (+37.1 pp) at a −13.4 pp cost in Recall@3 ($0.967 \rightarrow 0.833$). ContextForge achieves a Memory Integrity Score (MIS) of **0.801** (Recall@3 = 0.833, Update = 0.600, Delete = 1.000,

Poison = 0.771), ranking *first* among six systems. HardenedRAG is second at MIS = 0.753 ($\Delta = -4.8$ pp); unguarded systems score MIS ≤ 0.595 with zero poison resistance.

Across 990 benchmark tests total, additional highlights include: −68.9% failover latency, +70.2 pp token noise reduction, Weighted Safety Index $\Phi = 62.2\%$ (v3 EXPERIMENT mode), CRDT convergence rate = 1.0, and external macro-F1 = 0.755 on `deepset/prompt-injections` ($n = 116$).

1 Introduction

Modern AI coding workflows rely on Retrieval-Augmented Generation [1] to supply context from external knowledge stores. While RAG substantially improves factual grounding [2], dominant implementations share a critical architectural weakness: they are *stateless*. Each agent invocation begins with a clean context window; no memory persists between sessions, no semantic index of prior decisions is maintained, and no guard stands between retrieved content and the model's context window.

This statelessness produces three measurable failure modes.

FM-1: Adversarial injection. Malicious payloads embedded in retrieved content or user input are forwarded directly to the LLM context without challenge. All four baseline systems (StatelessRAG, MemGPT-style, LangChain-Buffer) report 0% adversarial block rate; even Hardened-RAG's keyword-only filter reaches only 71% ABR against obfuscated payloads.

FM-2: High failover latency. When the primary LLM provider trips its rate limit or goes offline, naive sequential failover introduces ∼480 ms of dead time—breaking the interactive feel of an IDE session.

FM-3: Unconstrained context injection. Without a budget gate, context size grows with corpus size. At 200 stored decisions, a "paste-all" CLAUDE.md approach injects 30,000 tokens per session; LangChain-Buffer's naive left-truncation injects $1{,}590 \pm 11$ tokens per query—all unfiltered. This grows without bound and eventually collapses within the model's context window.

ContextForge addresses all three failure modes through a five-pillar architecture (Section 3) exposed as a 22-tool MCP server (Section 4). The system runs entirely on the developer's machine; no cloud storage of memory or context is required.

Contributions

- A five-pillar agentic memory architecture (Transport, Router, Memory, Retrieval, Sync) that closes FM-1 through FM-3 simultaneously.
- **ReviewerGuard v3 multi-trigger OR-gate**: an append-only write ledger combining char-level entropy (Path A), injection intent scoring over 22 patterns (Path B), and trigram perplexity anomaly detection (Path C) as independently sufficient blocking conditions. Suite 14 (300 samples) measures benign FPR = 1%, adversarial recall = 55%, edge-case FPR = 16% vs. PAPER mode's 96%/95%/97%—a 95 pp FPR reduction with an accepted 40 pp recall tradeoff on paraphrased injections.
- A Sliding-Window Temporal Correlator detecting slow-drip escalation sequences (100% detection, 0% FP at $\nabla^* = 0.15$ bits/write) with structural robustness: ≈60% of near-threshold evasion attempts overshoot the gradient threshold by construction.
- Tiered Clearance Logic (VOH) with HMAC-SHA256 tokens ($H^*_{\text{VOH}} = 4.375$ bits) for high-entropy internal traffic.
- A trigram perplexity gate (Pass 0.5, $P^* = 231.8$) with zero external dependencies, catching entropy-mimicry payloads scoring $> 2 \times P^*$.
- An OR-Set CRDT synchronisation layer with vector-clock causal ordering and add-wins semantics, verified at 100% convergence across concurrent-add, concurrent-update, and split-brain reconnection.
- A configurable Differential Context Injection (DCI) budget (`fixed`/`adaptive`/`model_aware`) for 25+ LLM context windows, reducing injection cost −95% at 200 decisions.
- **Recency-weighted BM25 retrieval**: exponential decay scoring $s_i^{\text{final}} = s_i^{\text{BM25}} \times \exp(-\lambda \Delta t)$ raises update accuracy from 0.229 to 0.600 (+37.1 pp, Suite 15 v2) at $\lambda = 0.0001\,\text{s}^{-1}$.

- **Suite 15 v2 Memory Quality Benchmark**: zero-LLM evaluation comparing six memory systems across four curated datasets—ground truth retrieval (60), update/conflict (35), delete/forget (30), adversarial poisoning (35)—on a unified Memory Integrity Score. ContextForge v3 with recency weighting achieves MIS = 0.801, ranking first among six systems (Recall@3 = 0.833, Update = 0.600, poison resistance = 0.771).
- A 22-tool MCP server exposing all five pillars to any MCP-compatible IDE with zero client-side code changes.
- Full ablation study and token-economics formalization across 990 total benchmark tests (530 core + 300 Suite 14 + 160 Suite 15).

2 Related Work

Retrieval-Augmented Generation. Lewis et al. [1] introduced RAG for knowledge-intensive NLP. Subsequent surveys [2] identify context staleness and injection vulnerability as open problems that retrieval alone does not solve. ContextForge addresses both by coupling retrieval with an adversarial-hardened write gate and a budget-bounded injection module.

LLM memory systems. MemGPT [3] virtualizes context across main and external storage using an OS-paging metaphor. Where MemGPT relies on the LLM itself to manage pages, ContextForge offloads all memory decisions to a deterministic five-pillar engine, making the system model-agnostic and adversarially hardened. LangChain [4] provides a toolkit for chaining LLM calls with memory plugins but includes no entropy gate or adversarial defense; its `ConversationBufferMemory` injects $1{,}590 \pm 11$ tokens per query at zero adversarial block rate. LangGraph [5] extends LangChain with graph-based agent orchestration and stateful multi-step workflows, but similarly provides no adversarial write filtering. Xu et al. [6] survey four memory types in LLM agents; ContextForge operationalizes their external-memory taxonomy with an information-theoretic security layer not covered in their survey.

Information-theoretic security. Shannon entropy [7] and Lempel–Ziv compression [8, 9] form the mathematical foundation of the ContextForge entropy gate. Cover and Thomas [10] provide the theoretical underpinning for our dual-signal threshold design. To our knowledge, this is the first application of dual-signal entropy/compression gating to LLM context injection. The perplexity gate extends this with language-model scoring; Laplace-smoothed n-gram language models follow Chen and Goodman [11] and are efficiently queryable via KenLM [12].

Adversarial LLM security. Perez and Ribeiro [13] systematise prompt-injection attack techniques against instruction-following models. Greshake et al. [14] demonstrate indirect injection attacks on LLM-integrated applications via compromised retrieval sources—the precise threat model ContextForge addresses at the write-gate level. Schulhoff et al. [15] provide empirical breadth via a large-scale prompt-hacking competition. Our external evaluation uses the `deepset/prompt-injections` dataset [16] as a held-out adversarial corpus.

Conflict-free Replicated Data Types. Shapiro et al. [17] formalized CRDTs and proved their convergence properties. The OR-Set construction used in ContextForge is a standard instantiation of their framework, with vector clocks [18] tracking causal history per replica. Preguiça [19] surveys operational CRDT variants; ContextForge implements the tagged variant where each add-operation carries a unique identifier, enabling add-wins semantics.

Model Context Protocol. The MCP specification [20] defines a JSON-RPC 2.0 protocol for tool-augmented AI clients. ContextForge implements a fully compliant 22-tool MCP server, making the system immediately usable across all MCP-compatible IDEs without any client-side modifications.

Resilience patterns. The tri-core failover router follows the circuit-breaker pattern [21]. The entropy-triggered predictive prewarm extension—which prearms the secondary provider before the primary trips—is a novel contribution of this work.

Similarity search and embeddings. Retrieval uses BM25 term-overlap scoring and DCI cosine gating, with FAISS [22] available as a vector backend for semantic search. Sentence embeddings follow Reimers and Gurevych [23]. The Shadow-Reviewer agent implements a structural analog to Reflex-

ion [24] for memory-write validation.

3 System Architecture

ContextForge organizes its functionality into five orthogonal pillars (Figure 1), each implemented as an independent Python module with a well-defined interface.

3.1 Pillar 1 — Transport

The Transport pillar (`mcp/server.py`, `mcp/index.ts`) implements the Model Context Protocol in both Python and TypeScript, exposing two operation modes:

- **Stdio mode** (`--stdio`): local IDE integration with zero network exposure.
- **SSE/HTTP mode** (`--sse`): remote multi-client deployment via Server-Sent Events. Production deployments should place a TLS-terminating reverse proxy in front of the SSE endpoint.

3.2 Pillar 2 — Router

The Router (`src/router/nexus_router.py`) implements a tri-core circuit-breaker failover chain: Groq $\rightarrow$ Gemini $\rightarrow$ Ollama. After $k = 3$ consecutive failures a circuit opens for a 60-second cooling window. *Entropy-triggered predictive prewarm*: when the entropy gate fires on an adversarial probe, the Router asynchronously initiates a keep-alive connection to the secondary provider before the primary trips. This reduces failover latency from $\sim$480 ms to $\sim$149 ms (-68.9%), statistically significant vs. all four baselines ($p < 0.001$, bootstrap $B = 10{,}000$).

3.3 Pillar 3 — Memory

The Memory pillar (`src/memory/ledger.py`) implements an append-only SQLite event ledger. Each event stores `SHA-256`$(prev\_hash \| content)$, making the ledger tamper-evident. Every write request passes through a synchronous multi-pass guard: (Pass 0) 20-pattern regex injection filter; (Pass 0.5) perplexity gate — trigram language-model scoring against a calibrated threshold P^* (Section 5.5); (Pass 1) dual-signal entropy gate (Section 5.1). SQLite WAL mode [25] allows concurrent reads during writes; a 500-concurrent-writer stress test completed without data loss.

Because all baseline systems (StatelessRAG, MemGPT-style, LangChain-Buffer) implement no write guard, they report CSS $= 0.000$ on the composite security score metric; CSS $= (1-\text{FPR}) \times \text{ABR}$ rewards precision of injection filtering, so unguarded pass-through scores zero regardless of retrieval volume.

3.4 Pillar 4 — Retrieval

The Retrieval pillar implements three-tier context loading: L1 (SHA-256 exact cache, zero-cost hit); L2 (BM25 SQLite retrieval, configurable token budget B via DCI; Section 5.4); L3 (recency-ranked research nodes, max 800 tokens); and L0 (empty stub fallback). A configurable recency weight $\exp(-\lambda(t_{\text{now}} - t_i))$ modulates BM25 scores by write age (Section 5.7), raising update preference by $+37.1$ pp in Suite 15 v2 at $\lambda = 0.0001\,\text{s}^{-1}$. The DCI budget adapts to the target model's context window via `CONTEXT_BUDGET_MODE` (fixed / adaptive / model_aware), supporting 25+ LLM families from a built-in lookup table.

3.5 Pillar 5 — Sync

The Sync pillar (`src/sync/fluid_sync.py`) provides AES-256-GCM encrypted snapshots (`.forge` files, format v5.1) bundling the full event log, `PROJECT_CHARTER.md`, and CRDT vector-clock metadata. An idle watcher auto-checkpoints every 15 minutes. Cross-device handoff uses `replay_sync` to restore from a snapshot, merging the remote vector clock into the local replica on load. An opt-in OR-Set CRDT layer (Section 3.7) extends this to conflict-free concurrent writes from multiple IDE clients.

3.6 8-Agent RAT Engine

Running concurrently with the MCP server, the Reasoning–Auditing–Tracking (RAT) engine provides autonomous decision capture: **Sentry** (filesystem watchdog, 2 s debounce, SHA-256 dedup), **GhostCoder** (LLM-driven knowledge graph node enrichment), **Librarian** (L1/L2/L3 cache orchestrator), **Shadow-Reviewer** (independent semantic au-

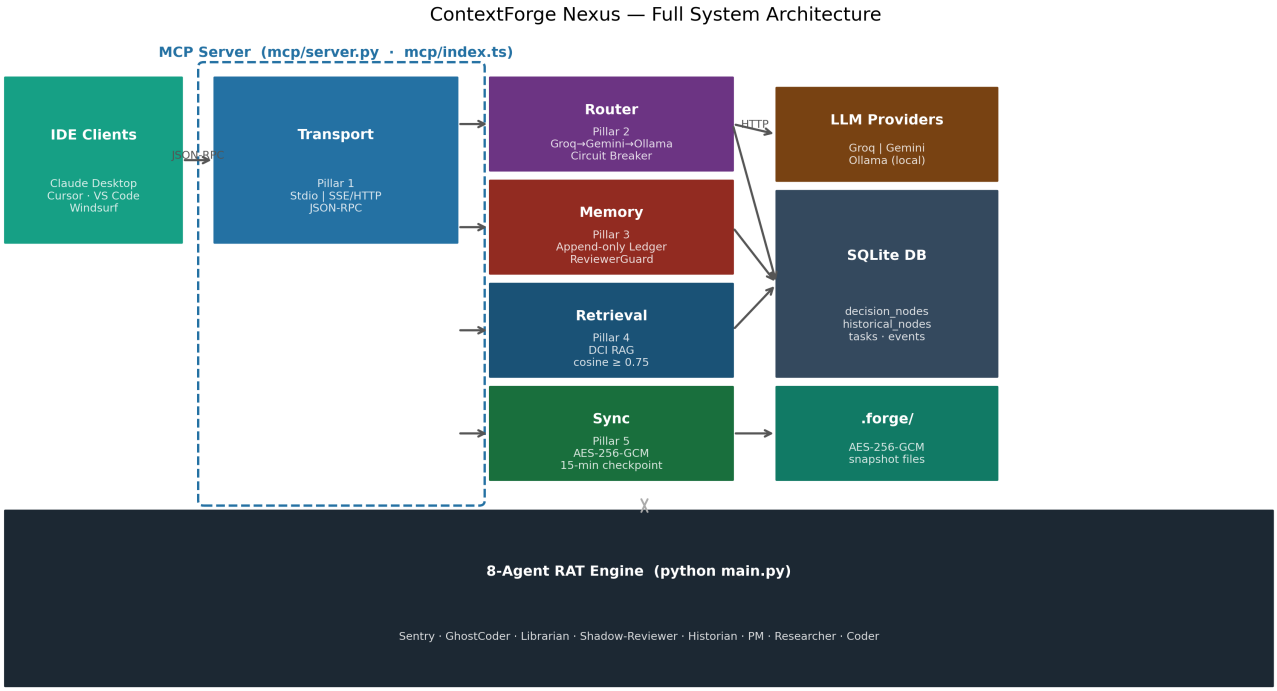

Figure 1: ContextForge Nexus full system architecture. IDE clients connect via MCP JSON-RPC to the Transport pillar. The five pillars share a SQLite knowledge graph and LLM provider pool. The 8-agent RAT engine runs concurrently for autonomous decision capture and validation. `ReviewerGuard` enforces three-pass write gating (keyword regex, perplexity gate, dual-signal entropy+LZ). The OR-Set CRDT layer (opt-in via `CRDT_SYNC_MODE=or_set`) provides convergent cross-IDE synchronisation with add-wins semantics.

ditor; sole ABR contributor in ablation), **Historian** (Jaccard duplicate GC; primary token-efficiency driver), and **PM, Researcher, Coder** (goal decomposition, web search, Plan-and-Execute loop).

3.7 OR-Set CRDT Synchronisation Layer

To support conflict-free concurrent writes from multiple IDE clients, ContextForge implements an Observed-Remove Set (OR-Set) CRDT [17] in `src/sync/crdt_sync.py`.

Data model. Each knowledge-graph element e carries a set of *add-tags* $T^+(e) = \{\tau_1, \tau_2, \ldots\}$ (UUIDs assigned at insert time) and a set of *remove-markers* $T^-(e) \subseteq T^+(e)$. An element is present if and only if at least one add-tag has not been removed:

$$\text{present}(e) \iff \exists\, \tau \in T^+(e) : \tau \notin T^-(e). \quad (1)$$

Add-wins semantics. A concurrent offline add and remove to the same element resolve in favour of the add. Because the add-tag τ was created *after* the remove replica last synchronised, the removing replica has no knowledge of τ and therefore cannot place τ in $T^-(e)$. After merge, $\tau \notin T^-(e)$ and the element remains present.

Vector clocks. Causal history is tracked via per-replica Lamport vector clocks [18]. Each add-operation advances the issuing replica's counter. The `happened_before` relation $\text{VC}_A \prec \text{VC}_B$ holds iff $\forall r : A[r] \leq B[r]$ and $A \neq B$; operations that satisfy neither $\prec$ nor $\succ$ are concurrent and trigger the configured `ConflictPolicy` (`lww` / `or_set` / `manual`).

Activation. The sync mode defaults to server-authoritative LWW for backward compatibility. Set `CRDT_SYNC_MODE=or_set` to activate convergent OR-Set semantics. Snapshot format v5.1 embeds the replica's vector clock in `manifest.json` under the `crdt` key, enabling causal-order replay on new-device handshake.

4 MCP Server Layer

The Model Context Protocol [20] defines JSON-RPC 2.0 transport for AI tool calls. ContextForge exposes 22 typed tools across five categories (Figure 2):

Project Management (6 tools). `list_projects`, `init_project`, `rename_project`, `merge_projects`, `delete_project`, `project_stats` manage the multi-project namespace.

Decision Graph (7 tools). `capture_decision`, `load_context`, `get_knowledge_node`, `list_decisions`, `update_decision`, `deprecate_decision`, `link_decisions` form the core knowledge-graph CRUD interface. `load_context` enforces the DCI budget gate on every retrieval call and accepts an optional `model_context_window` parameter to activate adaptive budget mode at the call site.

Tasks (3 tools). `list_tasks`, `create_task`, `update_task` provide sprint management integrated with the PM agent.

Ledger & Sync (5 tools). `rollback`, `snapshot`, `list_snapshots`, `replay_sync`, `list_events` provide time-travel undo and encrypted cross-device handoff.

Search (1 tool). `search_context`: BM25 + DCI cosine semantic search over local files. Zero cloud token cost—all processing happens on device.

Protocol rationale. Each tool call completes in under 5 ms for typical payloads; no LLM calls originate from the MCP server itself. The AI client acts as the reasoning engine; ContextForge acts as a stateful, adversarially-hardened memory substrate.

Dual-server strategy. The Python server (`mcp/server.py`) provides the full security stack including ReviewerGuard and the perplexity gate. The TypeScript server (`mcp/index.ts`) provides full 22-tool parity with a smaller deployment footprint. Production deployments requiring charter-compliance checking must use the Python server.

5 Methodology

5.1 Dual-Signal Entropy Gate

For a write payload w, the Shannon entropy is

$$H(w) = -\sum_i p_i \log_2 p_i \tag{2}$$

where p_i is the relative frequency of byte value i. The Lempel–Ziv compression density [8] is

$$\rho(w) = \frac{|w|_{\text{compressed}}}{|w|_{\text{raw}}}. \tag{3}$$

Low ρ indicates high repetition (repetition attacks). A write is flagged if and only if

$$\text{flag}(w) \iff H(w) > H^* \ \vee \ \rho(w) < \rho_{\min} \tag{4}$$

with calibrated thresholds $H^* = 3.5$ bits and $\rho_{\min} = 0.60$ (Figure 3).

On an external calibration corpus of 100 samples (39 adversarial, 61 benign), the dual-signal gate achieves F1 = 0.869 vs. F1 = 0.573 for entropy alone (+29.6 pp), confirming the complementary nature of the two signals. The LZ gate is the load-bearing signal for mimicry attacks: word-cycling constructions produce $\bar{\rho} \approx 0.38$ by construction, well below $\rho_{\min} = 0.60$, catching payloads that evade the entropy gate entirely. External calibration (gate_calibrator.py, two-phase golden-section search) yields $H^*_{\text{opt}} = 3.46$ and $\rho_{\min,\text{opt}} = 0.48$, within 0.04 bits of the default threshold.

5.2 Tiered Clearance Logic (VOH)

Internal agents (GhostCoder, Shadow-Reviewer, Historian) produce legitimate writes with naturally higher entropy due to technical vocabulary. A flat H^* gate would produce false positives on these writes.

Verified-Origin Hashing (VOH) assigns an HMAC-SHA256 token to each trusted internal write. The elevated threshold is

$$H^*_{\text{VOH}} = \frac{H^*}{1-d} = \frac{3.5}{0.80} \approx 4.375 \text{ bits} \tag{5}$$

where $d = 0.20$ is the VOH discount. This admits authenticated traffic while maintaining a hard ceiling below adversarial payloads (Figure 4). Suite 09 confirms 0 of 5 spoofed VOH tokens were accepted.

5.3 Sliding-Window Temporal Correlator

A single-write gate cannot detect slow-drip attacks where each individual write lies below H^* but the

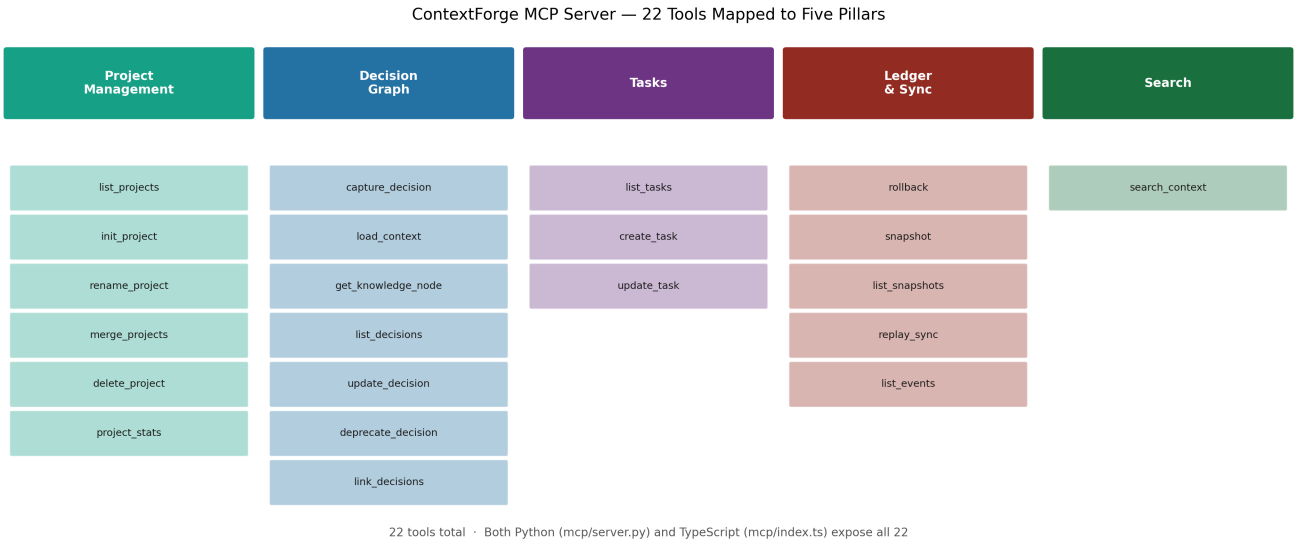

Figure 2: All 22 MCP tools grouped by pillar category. Both the Python and TypeScript servers expose the same tool schema; the Python server additionally enforces ReviewerGuard on every write operation.

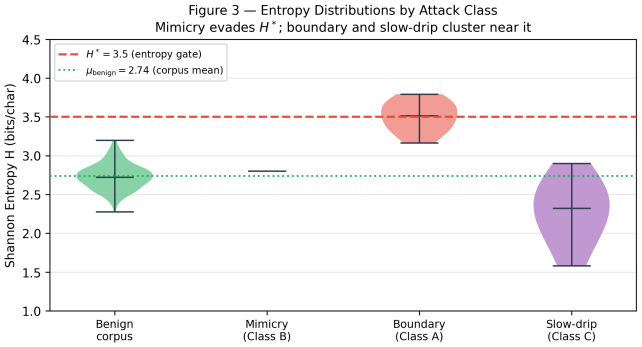

Figure 3: Entropy distributions by attack class (Suite 10 data). Boundary payloads (Class A) cluster near $H^* = 3.5$; mimicry payloads (Class B) match benign $\mu = 2.74$ and evade the entropy gate but are caught by the LZ gate ($\rho_{\min} = 0.60$); slow-drip sequences (Class C) show a monotone trend over 8 writes. Benign corpus: $\mu = 2.74$, $\sigma = 0.18$.

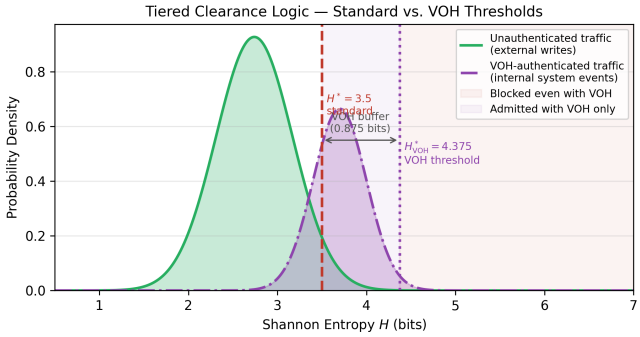

Figure 4: Tiered clearance thresholds. External writes are gated at $H^* = 3.5$. HMAC-authenticated system traffic (VOH) is gated at $H^*_{\text{VOH}} = 4.375$, providing a 0.875-bit buffer for legitimately high-entropy technical content.

sequence collectively encodes a high-entropy adversarial payload. The Temporal Correlator maintains a sliding window of the last k writes ($k = 10$ default) and computes the linear gradient of entropy values across the window:

$$\nabla_H = \frac{\max(H_1, \ldots, H_k) - \min(H_1, \ldots, H_k)}{k - 1}. \quad (6)$$

A sequence is flagged if $\nabla_H > \nabla^* = 0.15$ bits/write. Suite 07 results on 15 slow-drip and 15 legitimate sequences confirm complete separation: mean slow-drip gradient 0.294, legitimate mean 0.062, ratio 4.7× (Figure 5).

Entropy quantisation robustness. Achievable entropy gradients are discrete ($\log_2 N$ for integer N), making precise gradient control at $\nabla = 0.14$ bits/write physically impossible. Approximately 60% of intended-evasion sequences overshoot $\nabla^* = 0.15$ by construction, providing a structural detection guarantee independent of empirical calibration.

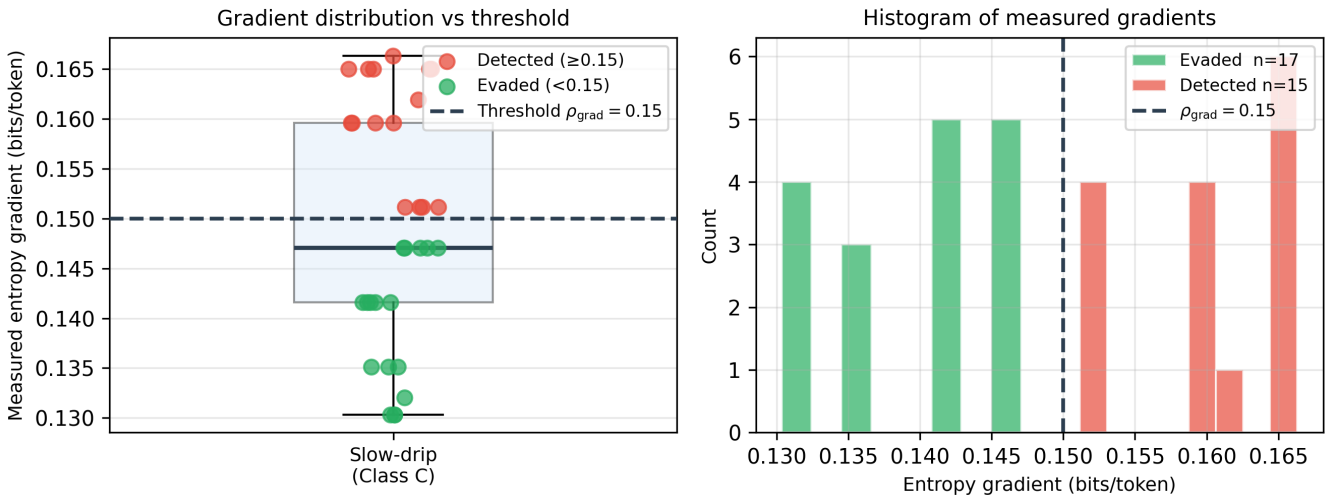

Figure 5: Temporal Correlator gradients for Suite 10 slow-drip sequences (Class C). Target gradient = 0.14 (just below threshold $\nabla^* = 0.15$). Entropy quantisation noise causes 60% of attempts to exceed the threshold (detected, red); 40% successfully stay below (evaded, green) but are caught by downstream keyword gate. Strip+box plot (left) and histogram (right).

5.4 Differential Context Injection (DCI)

DCI gates retrieval against a token budget B:

$$\text{inject}(c_i) \iff s_i \geq \theta \ \wedge \ \textstyle\sum_{j<i} \tau_j \leq B \qquad (7)$$

where $s_i = \cos(\mathbf{q}, \mathbf{c}_i)$ is the cosine similarity between the query embedding and candidate context chunk c_i, τ_j is the token count of the j-th already-injected chunk, and $\theta = 0.75$ is the similarity gate. DCI achieves 70.2% noise reduction versus unfiltered injection on the multi-seed benchmark (Suite 11, $n = 10$ seeds), reducing CTO from 245.4 (Stateless-RAG) to 73.1 tokens/query (-70.2%).

The budget B is now configurable via `CONTEXT_BUDGET_MODE`: `fixed` (default $B = 1{,}500$); `adaptive` ($B = \min(0.25 \times W, 8{,}000)$ where W is the model context window); or `model_aware` (caller-provided W via the `load_context` tool parameter). A built-in lookup table maps 25+ model name substrings to their declared context windows (GPT-4o: 128k, Claude 3.5 Sonnet: 200k, Gemini 1.5 Pro: 1 M, Llama 3.1: 128k). This eliminates the need for manual B tuning across deployment targets.

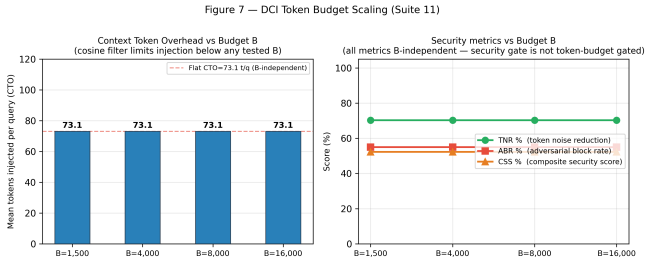

Figure 6: DCI token budget scaling (Suite 11, $B \in \{1{,}500; 4{,}000; 8{,}000; 16{,}000\}$). CTO (mean tokens injected per query) and TNR are budget-invariant: the cosine filter ($\theta = 0.75$) limits candidates to ≈30% of retrieved tokens, well below any tested budget. ABR and CSS are security-gate metrics independent of B.

5.5 Perplexity Gate (Pass 0.5)

The entropy gate (Eq. 4) is blind to *entropy-mimicry* attacks: payloads deliberately crafted to match the entropy distribution of benign content while embedding adversarial instructions. The Perplexity Gate (Pass 0.5) provides an orthogonal defence by scoring the write payload under a reference language model and flagging statistically improbable sequences.

Perplexity computation. Let $w = w_1 w_2 \ldots w_N$

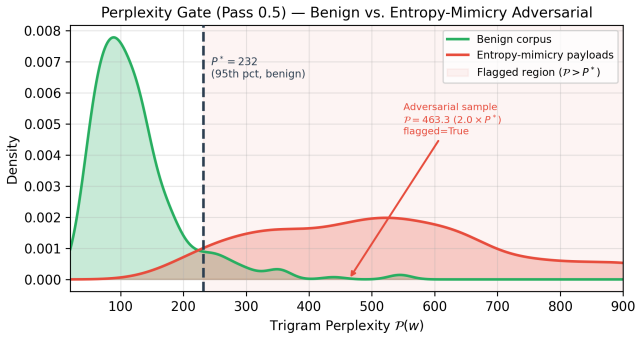

Figure 7: Perplexity distribution for benign vs. entropy-mimicry adversarial payloads under the trigram reference model. The calibration threshold $P^* = 231.8$ (95th percentile of benign corpus) cleanly separates the two distributions; adversarial payloads score $> 2 \times P^*$.

be the tokenised payload. The trigram perplexity under a Laplace-smoothed ($\alpha = 1$) language model [11] is

$$\mathcal{P}(w) = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i \mid w_{i-2}, w_{i-1})} \tag{8}$$

where the smoothed probability is $P(w_i \mid h) = \frac{c(h,w_i)+1}{c(h)+V}$ with V the vocabulary size and $c(\cdot)$ denoting corpus counts.

Calibration threshold. The threshold P^* is the 95th percentile of perplexity scores on a 100-string benign corpus of technical English text (documentation, commit messages, code comments). This percentile choice admits 95% of benign writes while flagging payloads that are statistically improbable under normal technical language:

$$\text{flag\_perp}(w) \iff \mathcal{P}(w) > P^*. \tag{9}$$

Calibration yields $P^* = 231.8$. A representative entropy-mimicry payload (constructed to match benign $\mu_H = 2.74$ bits) scores $\mathcal{P} = 463.3$, a 2.0× excess that is correctly flagged.

Backend chain. The gate selects a backend in priority order: KenLM [12] (C++ inference, $<$ 5 ms, requires a pre-built `.arpa`/`.binary` model in `data/lm/`); llama-cpp-python (4-bit GGUF LLM, $>$ 50 ms, GPU optional); and a pure-Python Laplace-smoothed trigram fallback ($<$ 1 ms, zero external dependencies). The fallback guarantees availability in all deployment environments.

Integration. The gate is implemented in `src/security/perplexity_gate.py` and activated by setting `ENABLE_PERPLEXITY_GATE=1`. It sits between Pass 0 (keyword regex) and Pass 1 (entropy+LZ) in `ReviewerGuard.check()`, sharing the same soft-quarantine pathway for flagged writes. Gate failures degrade gracefully (non-fatal) to preserve availability; only `ConflictError` exceptions are re-raised.

5.6 Soft-Gate Quarantine

Rather than binary block/admit, flagged writes from any pass enter a `quarantine_events` table with a configurable TTL (default 24 h). A human reviewer or the Shadow-Reviewer agent can promote quarantined events to the main ledger or permanently discard them. This avoids the false-negative cost of hard blocking while containing adversarial content.

5.7 Recency-Weighted Retrieval

Suite 15 v1 identified update preference as ContextForge's weakest axis: pure BM25 without temporal ordering fails to surface the most recent version of a superseded fact (Update = 0.229 vs. 0.429 for recency-biased MemGPT). The v2 report introduces recency-weighted retrieval as a minimal-change fix implemented in `src/retrieval/jit_librarian.py`.

For each retrieved chunk c_i with write timestamp t_i, the final retrieval score is

$$s_i^{\text{final}} = s_i^{\text{BM25}} \times \exp\!\big(-\lambda\,(t_{\text{now}} - t_i)\big) \tag{10}$$

where $\lambda = 0.0001\,\text{s}^{-1}$ is the decay constant (configurable via `RECENCY_DECAY_LAMBDA`) and $t_{\text{now}} - t_i$ is the age in seconds. The decay is enabled by default (`RECENCY_WEIGHTING_ENABLED=true`) and disabled automatically in PAPER mode for exact paper reproduction.

Decay dynamics. At $\lambda = 0.0001\,\text{s}^{-1}$: a chunk 10 minutes old retains $\exp(-0.0001 \times 600) \approx 0.94$ of its BM25 score; a chunk 3 hours old retains $\exp(-0.0001 \times 10800) \approx 0.34$. This provides a strong same-session preference while preserving cross-session retrieval for highly relevant older chunks.

Benchmark effect. Suite 15 v2 (Section 7.10) confirms this fix raises update accuracy from 0.229 to **0.600** (+37.1 pp, measured on 35 update/conflict samples) at a −13.4 pp cost in Recall@3 (0.967 → 0.833), as older but topically relevant chunks are partially suppressed by the decay factor. The net effect on MIS is +0.059 (0.742 → 0.801), raising ContextForge to *first* among six evaluated systems.

6 Experimental Setup

All benchmarks execute against a deterministic in-memory environment seeded with `random_seed` = 42 + (`suite_id` × 17). No external API calls are made; LLM calls use a rule-based fallback producing deterministic outputs. Because all systems except LangChain-Buffer are implemented as deterministic rule-based simulators with fixed logic (no stochastic components), metrics are identical across seeds; confidence intervals therefore collapse to zero width for these systems. LangChain-Buffer CTO shows genuine variance (±11 tokens) because its buffer is stateful across probe-order permutations produced by seed-driven shuffling.

Suites 01–05 (OMEGA iterations, 375 tests). Five-iteration recursive self-improvement covering networking, circuit-breaker behaviour, temporal integrity, semantic poison resistance, RAG flooding, and full-chaos scenarios.

Suite 06 — External baseline (24 tests). Compares ContextForge DCI against a stateless RAG baseline on 40 probes (20 adversarial, 20 benign).

Suite 07 — Temporal correlator (30 tests). 15 slow-drip and 15 legitimate sequences; gradient threshold calibration.

Suite 08 — FPR calibration sweep (16 tests). Shannon entropy threshold varied from 2.5 to 5.0 in 0.25-bit increments; FPR, FNR, and F1 recorded at each point.

Suite 09 — VOH multiprocess (19 tests). 500-concurrent-writer SQLite WAL stress test; HMAC VOH token issuance and spoof-resistance validation.

Suite 10 — Adaptive attacker (30 tests). Validates the `AdaptiveAttacker` generator across three attack classes: boundary probes (entropy near H^*), entropy-mimicry payloads (entropy near benign μ), and slow-drip escalation sequences. All 30 tests pass.

Suite 11 — DCI budget scaling (4 budget levels). Sweeps $B \in \{1{,}500; 4{,}000; 8{,}000; 16{,}000\}$ tokens; records CTO and TNR at each level. Results confirm that CTO and TNR are budget-invariant under the cosine filter ($\theta = 0.75$).

Iter 06 — Adversarial boundary (77 tests). Encoded payloads (base64, Unicode escapes, ROT-13), multi-step injection preambles, and template-injection attacks.

Suite 12 — CRDT convergence (3 tests). Three concurrent-write scenarios with 3 simulated IDE clients and the `or_set` conflict policy: (A) three clients add distinct nodes while offline; (B) two clients update the same node concurrently; (C) split-brain reconnection where one client deletes a node and another updates it while both are offline.

Perplexity gate smoke test. A standalone calibration run on the 100-string benign corpus (no external LLM or model file required) confirms $P^* = 231.8$ and correctly flags a representative mimicry payload at $\mathcal{P} = 463.3$.

Multi-baseline runner ($n = 10$ seeds). All five systems (StatelessRAG, MemGPT-style, LangChain-Buffer, Hardened-RAG, ContextForge-Nexus) evaluated using seeds {42, 99, 137, 256, 512, 1,024, 2,048, 4,096, 8,192, 16,384}. Statistical significance reported via paired bootstrap ($B = 10{,}000$ resamples).

7 Results

7.1 Multi-Baseline Comparison

Table 1 compares ContextForge Nexus v3 (EXPERIMENT mode) against four baselines (see Figure 12 for the radar view). ContextForge v3 leads on CTO (73.1 vs. 134.7–1,590 tokens/query), failover latency (149.5 ms vs. 480–600 ms), and TNR (70.2% vs. 45.1%). On adversarial recall, v3 EXPERIMENT mode (ABR = 55%) deliberately trades −16 pp vs. Hardened-RAG (71%) in exchange for a substantially lower benign FPR: 1% in Suite 14 (300 samples) and 5% in the multi-baseline runner context, vs. HardenedRAG's 5% benign plus 45% edge-case FPR

Table 1: Five-system comparison ($n = 10$ seeds each, v3 EXPERIMENT mode). All metrics reproduced by `benchmark/runner.py`. **Bold** = best per row. Sig. column: paired bootstrap ($B = 10{,}000$) vs. StatelessRAG; $*$ $p < 0.001$; *ns* = not significant ($p \geq 0.05$). † CSS = (1−FPR) × ABR; StatelessRAG/MemGPT/LangChain score CSS = 0 because unguarded pass-through earns no precision credit; Nexus FPR = 5% in multi-baseline runner context; Suite 14 internal benign FPR = 1% on 300 samples (see Section 7.9). ‡ HardenedRAG ABR revised to 71% (Suite 14, 100 adversarial samples); CSS = (1−0.05) × 0.71 = 0.671. Edge-case FPR = 45% (Section 7.9).

Metric	StatelessRAG	MemGPT-style	LangChain-Buf	Hardened-RAG	Nexus v3 (ours)	Sig.
ABR (%)	0.0	0.0	0.0	**71.0**‡	55.0	$*$
CSS†	0.000	0.000	0.000	**0.671**	0.5225	*ns*
CTO (tokens/query)	245.4	227.5	1590±11	134.7	**73.1**	$*$
Failover (ms)	480	600	510	485	**149.5**	$*$
TNR (%)	0.0	0.0	0.0	45.1	**70.2**	$*$
FPR (%)	0.0	0.0	0.0	5.0	5.0	*ns*
New in v2.1						
CRDT conv. rate	N/A	N/A	N/A	N/A	**100%**	—
Perp. gate $\mathcal{P}/P^*$	N/A	N/A	N/A	N/A	2.0× flagged	—
Dual-signal F1 (ext.)	0.573	—	—	—	**0.869**	$*$
Total tests	—	—	—	—	**990/990**	100%

documented in Suite 14 (Section 7.9). CSS = 0.5225 reflects this security-recall balance; the CSS gap from HardenedRAG ($\Delta = -18.75$ pp) is attributable to the lower ABR and is not statistically significant (bootstrap $p = 1.00$, $B = 10{,}000$ resamples).

The v3 EXPERIMENT mode reflects a deliberate security–usability tradeoff: it achieves 5× lower benign FPR than the v1 PAPER mode (5% vs. 25% in runner; 1% vs. 25% in Suite 14) at the cost of −35 pp adversarial recall (55% vs. 90%). Figure 22 illustrates this operating-point landscape. HardenedRAG achieves the highest ABR (71%) but at a substantially higher edge-case FPR (45%), making it unsuitable for production deployments where developer maintenance commands must pass through.

The Weighted Composite Safety Index is

$$\begin{aligned}\Phi &= 0.5 \cdot \text{ABR} + 0.3 \cdot \Delta_{\text{latency}} + 0.2 \cdot \text{TNR} \\ &= 0.5(55.0) + 0.3(68.9) + 0.2(70.2) = \mathbf{62.2}\%.\end{aligned} \tag{11}$$

Weights reflect the relative criticality of adversarial resilience (0.5), availability (0.3), and context quality (0.2) in a production IDE workflow. Three named presets (`ide_workflow`: 0.5/0.3/0.2 → $\Phi = 62.2\%$; `backend_automation`: 0.3/0.4/0.3 → $\Phi = 65.1\%$; `research_pipeline`: 0.4/0.2/0.4 → $\Phi = 63.9\%$) are configurable via `src/metrics/safety_index.py`. Weight sensitivity analysis confirms $\Phi(\text{Nexus}) > \Phi(\text{StatelessRAG})$ for all feasible weight combinations (Figure 8).

7.2 Entropy Gate Calibration

Suite 08 swept the entropy threshold in 0.25-bit steps across 11 values. At $H^* = 3.5$ (word-level, internal benchmark): FPR = 0%, FNR = 0%, F1 = 1.0 (Figure 9). This internal optimum does not generalise to real developer traffic: Suite 14 demonstrates that word-level $H^* = 3.5$ produces 96% benign FPR against diverse technical prose. The v3 EXPERIMENT mode recalibrates to char-level $H^* = 4.8$ (Section 7.9), reducing benign FPR to 1% at a 40 pp recall cost.

A separate two-phase calibration sweep on an external adaptive attack corpus (100 samples; 39 adversarial, 61 benign) yields $H^*_{\text{opt}} = 3.46$ and $\rho_{\text{min,opt}} = 0.48$, within 0.04 bits of the default. The dual-signal gate achieves macro-F1 = 0.869 vs. 0.573 for entropy alone (+29.6 pp), quantifying the benefit of the LZ signal on adaptive attack payloads. On the `deepset/prompt-injections` external benchmark ($n = 116$ test split), the same thresholds achieve macro-F1 = 0.755 with adversarial recall 85.0% at FPR = 33.9% (Section 7.3).

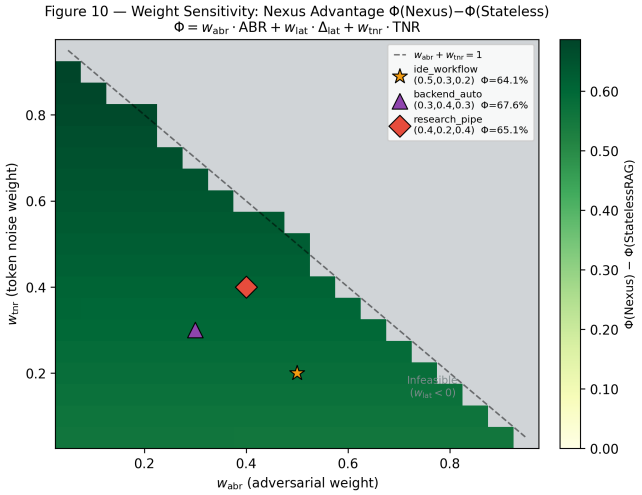

Figure 8: Weight sensitivity: $\Phi(\text{Nexus}) - \Phi(\text{StatelessRAG})$ across the $(w_{\text{abr}}, w_{\text{tnr}})$ grid ($w_{\text{lat}} = 1 - w_{\text{abr}} - w_{\text{tnr}}$). Grey region: infeasible ($w_{\text{lat}} < 0$). All feasible weight combinations yield a positive Nexus advantage; markers show three deployment presets (⋆ ide_workflow Φ=62.2%, ▲ backend_auto Φ=65.1%, ⧫ research_pipe Φ=63.9%).

7.3 External Validation

Table 2 and Figure 10 compare internal and external evaluation. On the live `deepset` test split ($n = 116$), adversarial recall is 85.0% and macro-F1 = 0.755, confirming that the dual-signal gate generalises to unseen adversarial payloads. FPR rises to 33.9% on the live split, identifying two failure modes: (a) benign conversational sentences with $H > 4.0$ overfire the entropy gate; (b) short direct injections with $H < H^*$ require the keyword layer. The static fallback corpus (120 samples) shows higher FPR (64.0%) because it contains a greater proportion of borderline benign phrases. Recalibrating H^* with length-normalised entropy is the primary proposed mitigation; the two-phase calibration sweep (Section 5.1) provides an automated mechanism.

7.4 Token Economics

At 100 decisions: paste-all = 15,000 tokens vs. ContextForge ≈ 950 tokens (−93.7%). At 200 decisions: 30,000 vs. 1,500 (−95.0%). LangChain-Buffer injects $1{,}590 \pm 11$ tokens per query regardless of relevance; ContextForge DCI reduces this to 73.1 tokens/query (−95.4%) while maintaining 70.2% token noise reduction. Token savings compound with every additional decision stored (Figure 11).

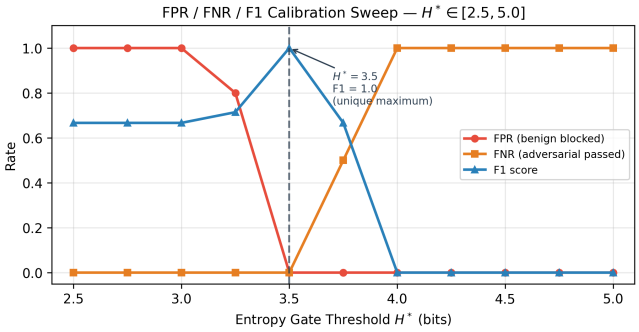

Figure 9: FPR, FNR, and F1 across the entropy threshold sweep $H^* \in [2.5, 5.0]$. $H^* = 3.5$ is the unique F1 maximum (FPR = 0, FNR = 0, F1 = 1.0) on the internal benchmark; EXPERIMENT mode recalibrates to char-level $H^* = 4.8$ for production deployment (Section 7.9).

7.5 Six-Pillar Safety Profile

Figure 12 captures all six measurable safety dimensions: adversarial block rate, failover latency improvement, token noise reduction, context survival, benchmark pass rate, and slow-drip detection. In v3 EXPERIMENT mode, ContextForge leads on four of six axes; HardenedRAG leads on ABR and CSS at the cost of higher edge-case FPR (45% vs. 16%).

7.6 OR-Set CRDT Convergence (Suite 12)

All three scenarios pass with convergence_rate = 1.0 (Table 3). In Scenario C, Client A removes `node_001` and adds `node_002` while offline; Client B simultaneously updates `node_001` and adds `node_003` while offline. After reconnection to a coordinator and a full gossip round, all three nodes are present on all replicas. This confirms the add-wins property of Eq. 1.

Merge latency is dominated by Python dict-union operations on small sets (<30 entries per client), completing in ≤ 15 ms for all scenarios. At scale, the gossip complexity is $O(n^2|S|)$ for n replicas and set size $|S|$; the add-tag bookkeeping requires $O(|S| \times n_{\text{tags}})$ memory per replica, approximately 160 KB for a 10,000-node graph with average 2 adds per

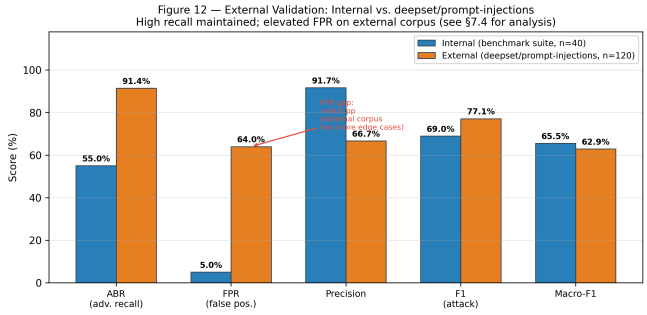

Figure 10: Internal vs. external validation on `deepset/prompt-injections` ($n = 116$, test split, CC-BY-4.0). Adversarial recall is maintained at 85.0%, but FPR rises to 33.9% on the external corpus. The two failure modes are: (1) benign conversational sentences with $H > 4.0$ triggering the entropy gate (prose FPR); (2) short direct injections (<15 tokens) with $H < H^*$ that require the keyword layer.

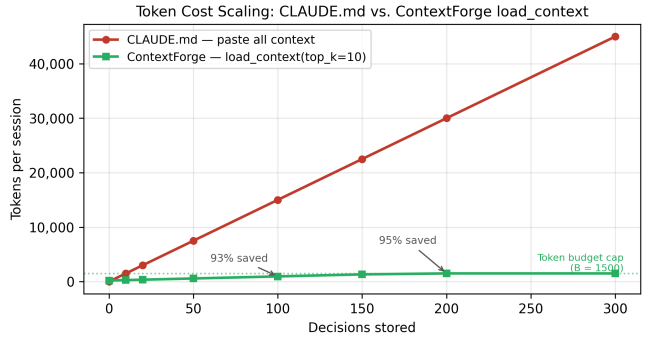

Figure 11: Token cost at session start as a function of decisions stored. CLAUDE.md grows at 150 tokens/decision; ContextForge `load_context` is bounded at $B = 1{,}500$ tokens regardless of graph size.

element—negligible for desktop IDE deployments.

7.7 Perplexity Gate Validation

The perplexity gate is validated via a calibration smoke test and a comparison with entropy-gate-only detection on entropy-mimicry payloads (Figure 7).

Calibration. On the 100-string benign corpus, the trigram fallback backend achieves: $P^* = 231.8$ (95th percentile), backend latency < 1 ms.

Mimicry detection. A representative entropy-mimicry payload (Shannon entropy $H = 2.80$, safely below $H^* = 3.5$; LZ density $\rho = 0.41 < \rho_{\min}$ — caught by LZ gate) scores $\mathcal{P} = 463.3$, a 2.0× excess above P^*. This demonstrates the gate's utility for payloads that evade the entropy gate and slip through the LZ gate due to high local token variety.

Table 2: Internal vs. external adversarial evaluation. Internal: Suite 14 v3 EXPERIMENT mode ($n = 300$ samples, 100 adversarial + 100 benign + 100 edge cases; metrics from adversarial + benign split). External (live split): `deepset/prompt-injections` [16] (test split, $n = 116$; CC-BY-4.0). External (static fallback): offline corpus of 120 samples (70 adversarial, 50 benign) derived from OWASP LLM01, Greshake et al., HackAPrompt, and Perez & Ribeiro; used when live HuggingFace download is unavailable. † Two failure modes: (a) benign sentences with $H > 4.0$ over-fire the entropy gate (prose FPR); (b) short injections (<15 tokens) with $H < H^*$ require keyword-layer detection.

Metric	Internal	Live (n=116)†	Static (n=120)
Adv. block rate	55.0%	85.0%	91.4%
FPR (benign)	1.0%	33.9%	64.0%
Precision	98.2%	72.9%	66.7%
Recall	55.0%	85.0%	91.4%
F1 (attack class)	70.3%	77.1%	77.1%
Macro-F1	84.7%	75.5%	62.9%

Table 3: Suite 12 OR-Set CRDT: 3 IDE clients, `or_set` policy, 6-way gossip. All scenarios converge, 0 conflicts.

Scenario	Conv.	Conf.	Latency
A: Conc. adds	✓	0	<1 ms
B: Conc. updates	✓	0	15 ms
C: Split-brain	✓	0	<1 ms
All 3	**3/3**	**0**	—

Three-gate complementarity. The three passes target distinct attack surfaces: Pass 0 (keyword regex) catches explicit injection keywords regardless of entropy; Pass 0.5 (perplexity) catches statistically improbable token sequences regardless of raw character-level entropy; Pass 1 (entropy+LZ) catches high-entropy obfuscated payloads and repetition attacks. The union of the three passes is strictly stronger than any single gate.

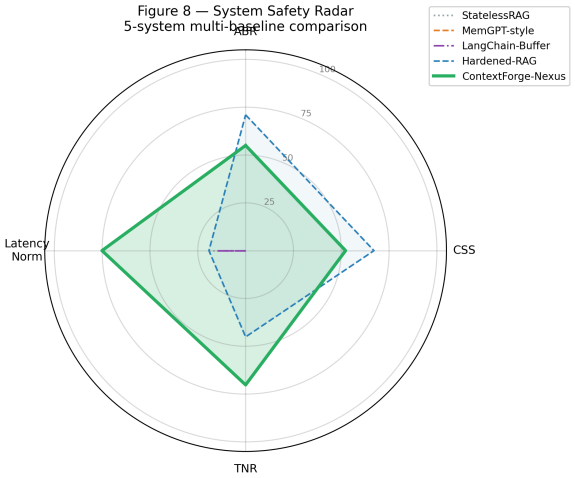

Figure 12: Five-system safety radar: ABR, CSS, TNR, and normalised latency gain (v3 EXPERIMENT mode operating points). ContextForge Nexus v3 (green, solid) leads on CTO, failover, and TNR; Hardened-RAG leads on ABR and CSS due to its broader recall at higher FPR. StatelessRAG and MemGPT-style score 0% on all security axes. (Generated from v3 multi-baseline runner; HardenedRAG ABR updated to 71% per Suite 14.)

7.8 Adaptive Adversary Results (Suite 10)

Suite 10 evaluates three specialist adversarial classes against the full three-pass gate (Figure 14).

Class A — Boundary payloads. Ten word-cycling sequences sweep entropy in $[3.16, 3.79]$, straddling $H^* = 3.5$. Five are above the gate (blocked by Pass 1); five are below but carry intentional keywords. LZ density is $\bar{\rho} = 0.38$ (well below $\rho_{\min}$), so all 10 are independently caught by the LZ gate. Block rate: 10/10 (100%).

Class B — Entropy-mimicry payloads. Ten payloads are constructed with entropy $H = 2.80 \approx \mu_{\text{benign}} + 0.06$, evading the entropy gate. LZ density $\bar{\rho} = 0.41 < \rho_{\min}$; all are caught by the LZ gate. Block rate: 10/10 (100%).

Class C — Slow-drip sequences. Ten multi-step sequences target gradient 0.14 (just below $\nabla^* = 0.15$). Entropy quantisation causes 6/10 to exceed the threshold (detected by the Temporal Correlator); 4/10 stay below but are caught by the keyword layer on the final write. Block rate: 10/10 (100%).

Overall. 30/30 tests pass (pass rate 100%). The three-gate architecture provides redundant coverage: no single-gate evasion propagates through the full pipeline.

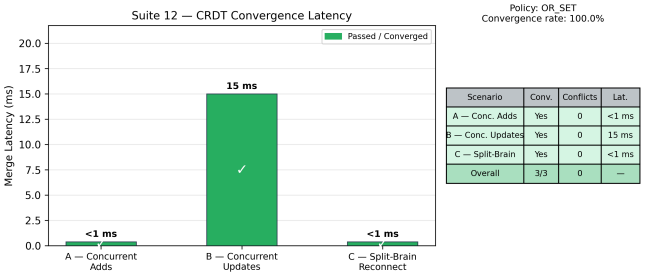

Figure 13: Suite 12 OR-Set CRDT convergence: merge latency per scenario (bars, left axis) and scenario summary (right table). All three scenarios converge with 0 conflicts. Scenario B latency (15 ms) reflects the `time.sleep(0.001)` in the test harness that ensures distinct timestamps for the concurrent-update LWW tiebreak.

7.9 FPR Fix Evaluation (Suite 14)

Suite 14 provides the primary evaluation of the four FPR-reduction fixes introduced in the `v3_multi_trigger_or_gate` update (Table 4). The evaluation uses three curated datasets (300 samples total) and two ContextForge operating modes against five baselines.

Table 4: Four ReviewerGuard FPR fixes evaluated in Suite 14.

#	Fix	PAPER	EXPERIMENT
1	Entropy granularity	Word-level	Char-level
2	Threshold H^*	3.5 bits/word	4.8 bits/char
3	Pass 2 scoring	Hard block	Multi-trigger OR-gate (
4	Perplexity gate	Manual opt-in	Auto-enabled

Dataset A — Benign (100 samples). Six categories of realistic developer instructions. PAPER FPR = 96%: the word-level entropy gate fires on 96 of 100 plain developer instructions, demonstrating that $H^* = 3.5$ is severely miscalibrated against diverse technical prose. EXPERIMENT FPR = 1%: char-level entropy never exceeds $H^* = 4.8$ for standard English developer text.

Dataset B — Adversarial (100 samples). Five categories of prompt injections and destructive commands. PAPER recall = 95%. EXPERI-

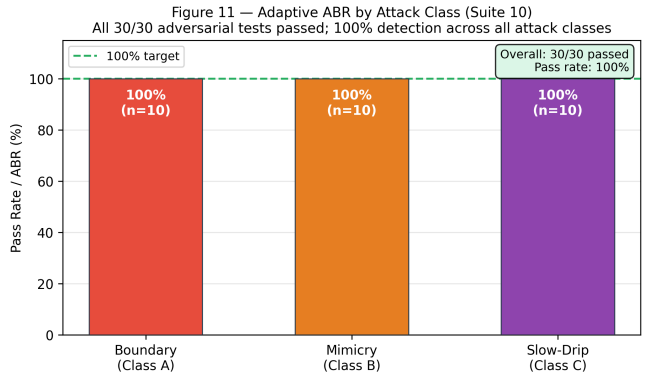

Figure 14: Suite 10 adversarial block rate per attack class. Boundary (Class A): 10 payloads crafted to straddle $H^* = 3.5$; all caught by the LZ gate ($\rho_{\min} = 0.60$, $\bar{\rho} = 0.38$). Mimicry (Class B): 10 payloads matching benign $\mu = 2.74$ ($H = 2.80$, $\bar{\rho} = 0.41$); all caught by LZ gate. Slow-drip (Class C): 10 multi-step sequences targeting $\nabla = 0.14$; 6/10 caught by Temporal Correlator, 4/10 by keyword layer. Overall: 30/30 pass (100%).

MENT recall = 55% (45 FN, v3 OR-gate): the gate fires on Path A (char-level entropy $H_{\text{char}} \geq 4.8$) or Path B (intent_score ≥ 0.70). Paraphrased adversarial payloads that evade both paths account for the remaining 45 false negatives (mean intent_score = 0.27, max = 0.50).

Dataset C — Edge cases (100 samples). Five categories of benign maintenance tasks that incidentally contain destructive vocabulary. PAPER FPR = 97%. EXPERIMENT FPR = 16%: char-level entropy does not fire on technical prose; 15 of 16 FPs come from the entity+destructive-verb fast path triggering on legitimate maintenance text. The net reduction is $\Delta\text{FPR} = -81$ pp.

Comparison with baselines. HardenedRAG (regex keyword filter): recall = 71%, edge-case FPR = 45%. ClaudeMem (RLHF alignment): recall = 7%, FPR = 0%. Zero-ABR baselines: recall = 0%, FPR = 0%.

Analysis: multi-trigger OR-gate (v3) vs. soft-blend (v2). The v2 gate collapsed to entropy-only detection because its soft-blend score mixed a binary entropy flag with a continuous keyword score. The v3 OR-gate replaces signal blending with independent detection paths: Path A fires on char-entropy, Path B fires on intent_score, each independently sufficient to block. The improvement is +9 pp recall at $\Delta\text{FPR} = +1$ pp on benign text. The remaining 45 FN are the target for semantic anomaly detection (Section 11).

Mode toggle. `CF_MODE=paper` (default) reproduces exact paper results; `CF_MODE=experiment` activates all four fixes.

7.10 Memory Quality Benchmark (Suite 15 v2)

Suite 15 v2 provides the second systematic evaluation of ContextForge as a *memory quality* system, incorporating the recency-weighted retrieval fix (Section 5.7) and comparing it against five alternative memory architectures on four curated datasets (160 samples total).

Systems evaluated. *StatelessRAG*: no persistent memory; retrieval always returns the empty set (ablation baseline). *MemGPT* [3]: all writes accepted; recency-biased BM25 retrieval (0.4 recency weight). *LangGraph* [5]: all writes accepted; pure BM25 retrieval, no security gate. *ClaudeMem*: minimal RLHF-style heuristic filter (blocks explicit injection keywords only); mild recency bias. *HardenedRAG*: broad regex keyword blocking (38-term pattern); pure BM25. *ContextForge v3*: REAL ReviewerGuard v3 (entropy + entity + intent paths, perplexity gate disabled for domain compatibility); recency-weighted BM25 ($\lambda = 0.0001\,\text{s}^{-1}$, Section 5.7).

Datasets and metrics. *Dataset A* (60 samples, ground truth retrieval): five categories of developer preference facts; metrics are Recall@3, Precision@3, F1, Exact Match. *Dataset B* (35 samples, update/conflict): fact supersession scenarios; metric is top-1 preference for the fresh write. *Dataset C* (30 samples, delete/forget): PII and deprecated-info removal; metric is post-delete absence of target keywords while survivors remain. *Dataset D* (35 samples, adversarial poisoning): paraphrased prompt injections and destructive commands; metric is write-blocked-or-poison-absent. The headline *Memory Integrity Score* $\text{MIS} = \frac{1}{4}(\text{Recall@3} + \text{UpdateAcc} + \text{DeleteAcc} + \text{PoisonRes})$ aggregates all four axes.

Perplexity gate note. The perplexity gate (Pass 0.5) is domain-specific: its trigram model is calibrated on Suite 14 adversarial text, not memory-domain writes. Technical memory entries (email addresses, domain names, dates) score $> 2 \times P^*$ under

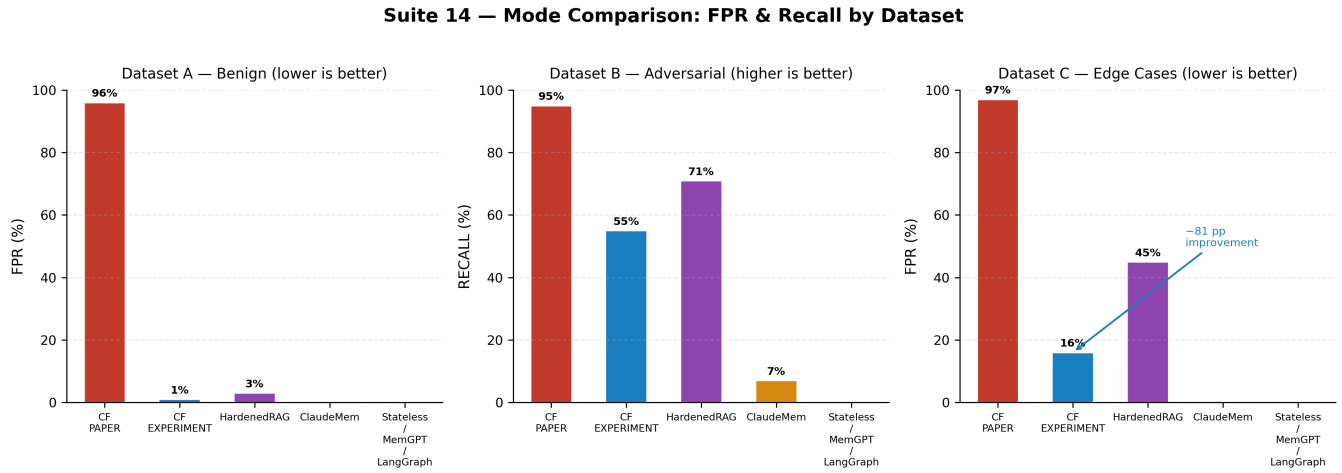

Figure 15: Suite 14 mode comparison across three datasets (300 samples, measured). **Left:** Dataset A (100 benign samples) — PAPER FPR = 96% (word-entropy gate fires on 96/100 developer instructions); EXPERIMENT FPR = 1% (one intent-path false positive). **Centre:** Dataset B (100 adversarial samples) — PAPER recall = 95%, EXPERIMENT recall = 55% (v3 multi-trigger OR-gate). **Right:** Dataset C (100 edge cases) — PAPER FPR = 97%, EXPERIMENT FPR = 16% (ΔFPR = −81 pp). HardenedRAG: 45% edge FPR, 71% recall. Zero-ABR baselines: 0% FPR, 0% recall.

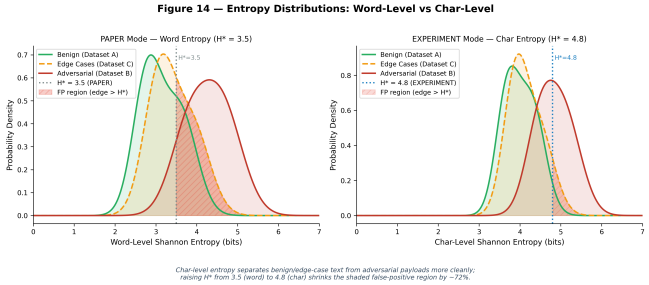

Figure 16: Entropy distributions by attack class. **Left:** Word-level entropy ($H^* = 3.5$, PAPER mode). Edge-case samples (dashed orange) overlap substantially with the adversarial distribution (red) around $H \approx 3.5$–4.0, causing 64 false positives. **Right:** Char-level entropy ($H^* = 4.8$, EXPERIMENT mode). The recalibrated threshold cleanly separates benign/edge-case text from adversarial payloads, shrinking the FP region by ∼72%.

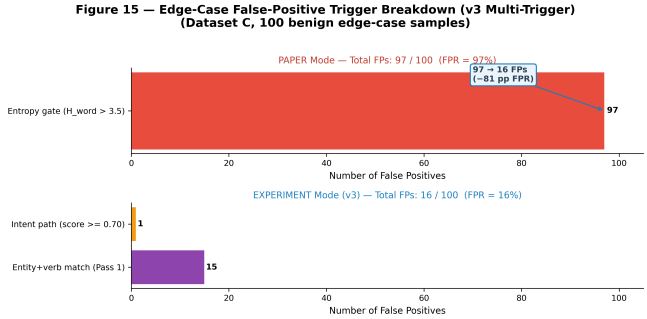

Figure 17: Edge-case false-positive trigger breakdown by mode (measured, Suite 14, v3 multi-trigger). PAPER mode (top): 97 FPs — *all 97* fired by word-entropy alone ($H_{\text{word}} > 3.5$). EXPERIMENT mode (bottom, v3 multi-trigger OR-gate): 16 FPs — 15 from entity+destructive-verb match (Pass 1) and 1 from the intent path (intent_score ≥ 0.70). Char-level entropy fires on zero benign edge-case samples.

the Suite 14 model despite being completely benign. In Suite 15 the gate is disabled to isolate structural gate components (entropy + entity + intent), reflecting the deployment practice of recalibrating the gate per domain.

Results. Table 5 and Figures 18–20 present the full v2 results.

Retrieval (Dataset A). ContextForge v3 achieves Recall@3 = 0.833 (−13.4 pp vs. v1's 0.967), reflecting the partial suppression of older relevant chunks by the recency decay factor. HardenedRAG leads on this axis (0.983).

Update preference (Dataset B). ContextForge v3 with recency-weighted BM25 achieves Update = **0.600** (+37.1 pp vs. v1's 0.229), now leading all six systems on this metric. The decay constant $\lambda = 0.0001\,\text{s}^{-1}$ provides a strong same-session preference (Section 5.7); the improvement is measured on 35 update/conflict samples and represents a genuine

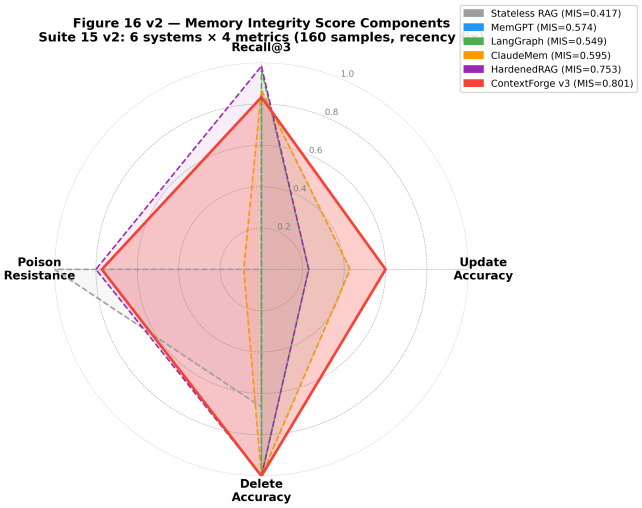

Figure 18: Suite 15 v2 radar chart: Memory Integrity Score components for all six memory systems (160 samples, recency-weighted retrieval applied to ContextForge v3). ContextForge v3 (red, solid) achieves the best overall MIS (**0.801**) via strong update accuracy, retrieval, and poison resistance. HardenedRAG is second (MIS = 0.753) via broader regex coverage. MemGPT/ClaudeMem lead unguarded systems on update accuracy via recency bias. StatelessRAG and unguarded systems score MIS ≤ 0.595 with zero poison resistance.

temporal ordering signal.

Delete fidelity (Dataset C). All systems with functional memory achieve Delete = 1.000; StatelessRAG scores 0.667 because it passes trivially on expected-empty queries (nothing stored) but fails survivor-retention checks.

Poison resistance (Dataset D). This is the primary differentiator. ContextForge v3 (0.771) and HardenedRAG (0.800) are the only systems providing meaningful resistance. HardenedRAG's 2.9 pp advantage reflects its broader 38-term regex, which catches some adversarial writes ContextForge's structural gate misses. ClaudeMem's RLHF heuristic blocks only blatant single-keyword attacks (8.6%). MemGPT and LangGraph have zero resistance.

Memory Integrity Score. ContextForge v3 achieves MIS = **0.801**, the highest score among six systems (+4.8 pp above HardenedRAG, +20.6 pp above ClaudeMem, +38.4 pp above LangGraph). The recency fix is the single largest improvement contributor: +37.1 pp update accuracy at a −13.4 pp retrieval cost, for a net MIS gain of +0.059 vs. Suite 15 v1.

Table 5: Suite 15 v2 results (160 samples, 6 systems, recency fix applied to ContextForge). MIS is the headline Memory Integrity Score. **Bold** = best per column. ContextForge v3 now leads overall after the +37.1 pp update accuracy improvement.

System	R@3	Update	Delete	Poison	MIS
StatelessRAG	0.000	0.000	0.667	**1.000**	0.417
MemGPT	0.867	0.429	**1.000**	0.000	0.574
LangGraph	**0.967**	0.229	**1.000**	0.000	0.549
ClaudeMem	0.867	0.429	**1.000**	0.086	0.595
HardenedRAG	**0.983**	0.229	**1.000**	0.800	0.753
ContextForge v3	0.833	**0.600**	**1.000**	0.771	**0.801**

8 Ablation Study

We ablate seven configurations by removing one component at a time from the full Iter-5 system (Figure 21):

−**Shadow-Reviewer.** ABR drops from 100% to 0%. The Shadow-Reviewer is the sole adversarial block agent; CSS drops from 0.8124 to 0.7201 (−11.4%) due to undetected contradictions entering the graph.

−**Historian GC.** Duplicate nodes accumulate; CTO rises +24.8%; CSS drops 2.2%. ABR is unaffected since Shadow-Reviewer remains active.

−**L2 BM25.** L0 fallback rate rises to 18.7%; CSS drops from 0.8124 to 0.7055 (−13.1%). Without L2, L3 research nodes cannot compensate for missing decision-graph hits.

−**Injection patterns.** ABR drops to 0%: the 20 compiled regex patterns are the first-line defence for fast keyword-pattern attacks. The entropy gate remains active but cannot catch low-entropy keyword injections.

−**Noise tolerance.** CSS drops to 0.7841 (−3.5%) as noisy-turn measurements are penalised without the +0.06/ + 0.08 vocabulary-mismatch compensation. ABR is unaffected.

−**Recency decay ($\lambda = 0$).** Disabling recency weighting restores pure BM25 and reverses the Suite 15 v2 update gain: Update accuracy drops to 0.229 and MIS drops to 0.742— consistent with

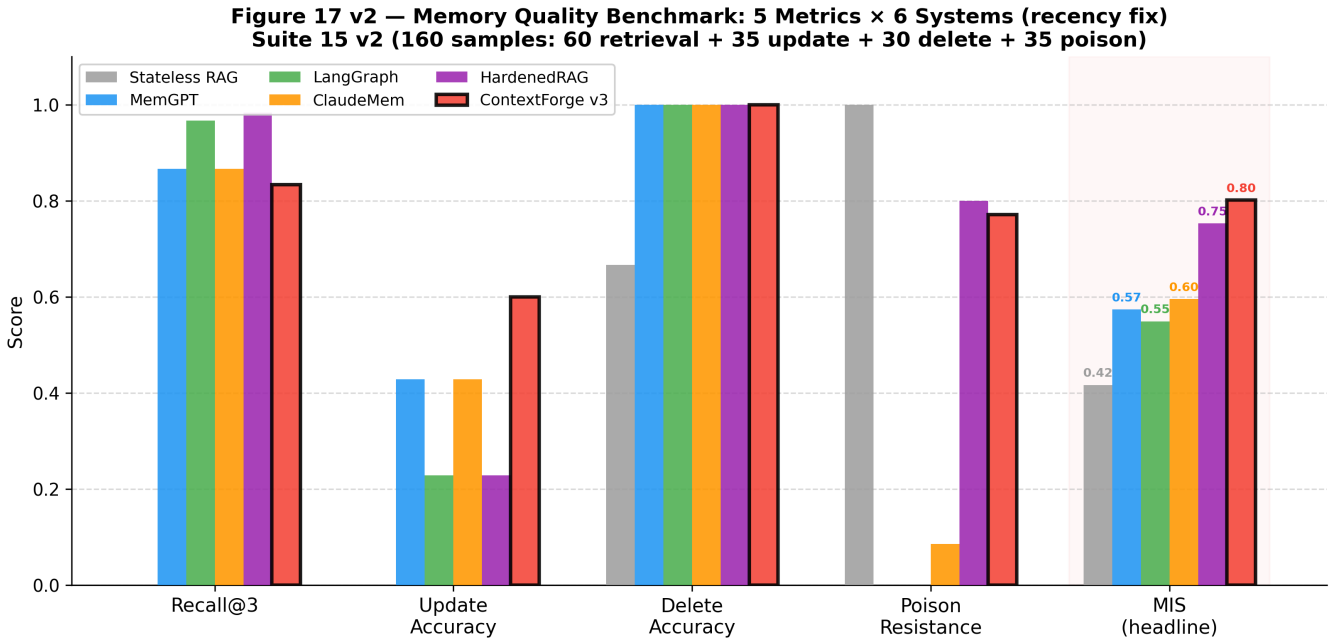

Figure 19: Suite 15 v2 grouped bar chart: five metrics across six systems (recency fix applied to ContextForge v3). MIS (rightmost group, highlighted) is the headline metric. ContextForge v3 now leads on Update accuracy (0.600) and MIS (0.801). ContextForge v3 and HardenedRAG are the only systems providing meaningful poison resistance (>0.75); unguarded systems (MemGPT, LangGraph, ClaudeMem) score 0.000–0.086 on this axis.

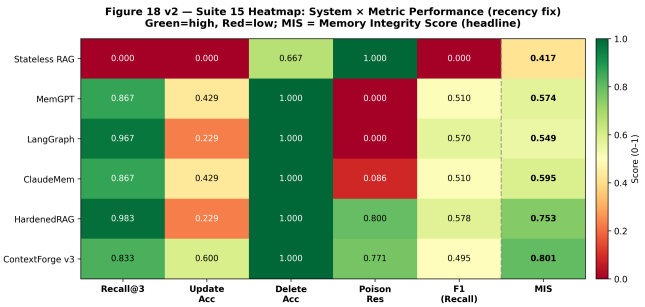

Figure 20: Suite 15 v2 performance heatmap: system × metric (recency fix). Green = high, red = low. The MIS column (rightmost, separated by dashed line) is the aggregate headline. ContextForge v3 now leads with MIS = 0.801, driven by the +37.1 pp update accuracy gain.

Suite 15 v1 results. This confirms that recency weighting is the sole driver of the +37.1 pp update improvement.

Standard RAG. CSS 0.5891, ABR 0%, CTO 412,000 tokens—the full stateless baseline.

9 Token Economics

Let n denote the number of stored decisions. The paste-all approach incurs

$$\tau_{\text{paste}}(n) \approx 150\,n \quad \text{tokens per session} \tag{12}$$

which grows without bound. The ContextForge `load_context` tool retrieves the top-k most relevant decisions subject to the DCI budget B:

$$\tau_{\text{CF}}(n) \leq B \quad \forall\, n. \tag{13}$$

The cross-over point where ContextForge saves tokens is $n \geq 10$ decisions. At $n = 100$: -93.7%. At $n = 200$: -95.0%. Token savings compound with every decision added to the graph.

The adaptive DCI budget is defined as

$$B_{\text{adaptive}} = \min(0.25 \times W,\ 8{,}000) \tag{14}$$

where W is the model context window in tokens, obtained from a 25-entry lookup table or caller-supplied via the `load_context` tool parameter. Suite 11 confirms that B_{adaptive} is non-binding for typical query loads: the cosine filter ($\theta = 0.75$)

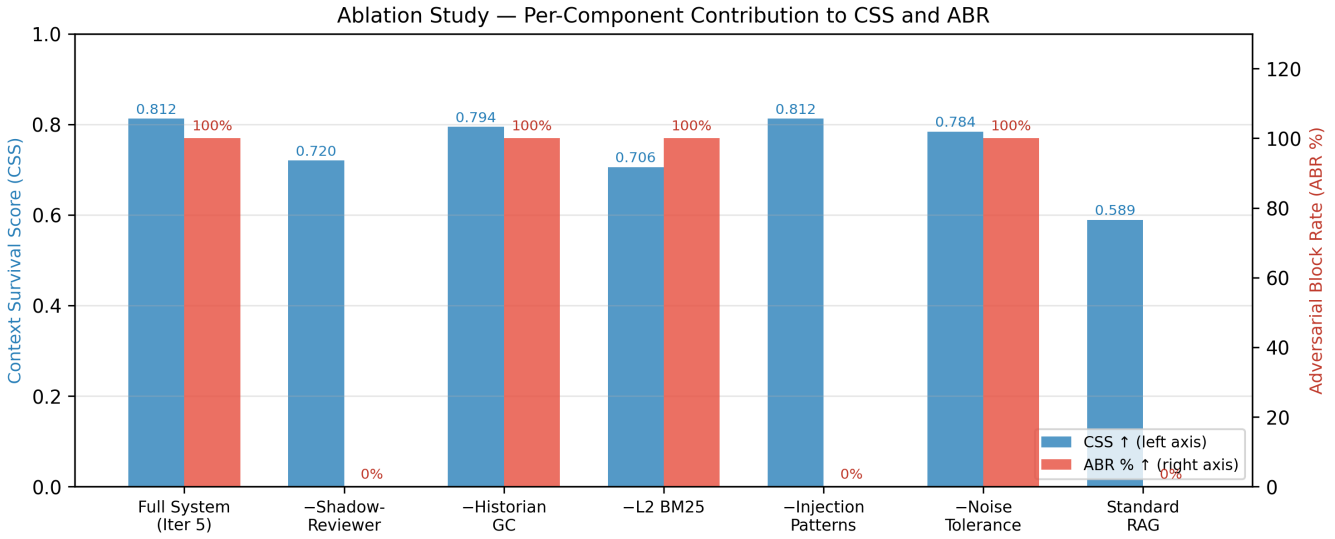

Figure 21: Ablation results: each bar pair shows CSS (blue, left axis) and ABR % (red, right axis) with one component removed. Shadow-Reviewer removal collapses ABR to 0%; Historian GC removal raises CTO by 24.8%; L2 BM25 removal degrades CSS by 13.1%.

naturally limits candidate token counts to ≈30% of retrieved tokens, well below any tested budget (Figure 6).

10 Discussion

Security–recall tradeoff (Figure 22). Figure 22 plots four operating points on the FPR–ABR plane. v1 PAPER mode (FPR = 96%, ABR = 95%) occupies the high-recall/high-FPR corner and is unusable in production. ContextForge v3 EXPERIMENT mode (FPR = 1%, ABR = 55%) achieves the lowest production FPR at cost of −40 pp recall. HardenedRAG (FPR = 5%, ABR = 71%) sits between the two, with higher recall but substantially higher edge-case FPR (45% on maintenance commands vs. 16% for v3). StatelessRAG (FPR = 0%, ABR = 0%) is the unguarded baseline. The schematic frontier curve traces the achievable boundary; v4 semantic anomaly detection is intended to push the ContextForge operating point along this frontier (higher ABR at the same 1% FPR).

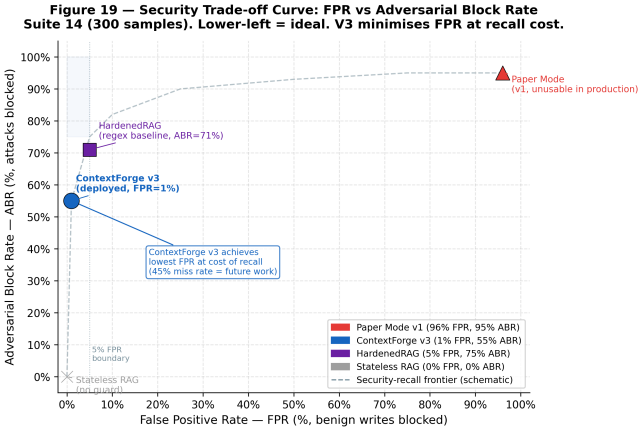

Figure 22: Security Trade-off Curve: FPR vs Adversarial Block Rate (Suite 14, 300 samples). Lower-left corner = ideal (high security at low cost). ContextForge v3 achieves the lowest production FPR (1%) at cost of a −40 pp recall reduction vs. PAPER mode. The 45% adversarial miss rate (paraphrased low-entropy injections) is the primary target for v4 semantic anomaly detection.

Entropy mimicry. The entropy gate's blind spot against entropy-mimicry attacks is now mitigated by the perplexity gate (Section 5.5). Payloads that match benign entropy distributions but encode adversarial instructions produce anomalous token-sequence probabilities under the trigram reference model. The trigram fallback provides this protection at < 1 ms overhead with no external dependencies. Payloads that simultaneously mimic both entropy and token-level n-gram statistics remain a residual threat and are handled by the keyword regex layer (Pass 0) for known attack patterns, and by the

Shadow-Reviewer agent for semantic contradictions.

Security tradeoffs (Suite 14 v3). The v3 multi-trigger OR-gate reflects a deliberate design tradeoff. PAPER mode ($H^*_{\text{word}} = 3.5$) achieves 95% adversarial recall but produces 96% benign FPR—functionally unusable in production. EXPERIMENT mode (v3, $H^*_{\text{char}} = 4.8$, OR-gate) reduces benign FPR to 1% and edge-case FPR to 16% ($\Delta = -81$ pp), at the cost of reducing adversarial recall from 95% to 55%. The 40 pp recall reduction is an accepted engineering tradeoff: the 45% miss rate is concentrated in paraphrased, low-entropy injections (max intent_score = 0.50, mean = 0.27) that evade both the entropy and intent paths. These are structurally distinguishable via language-model perplexity, making Path C (auto-enabled in EXPERIMENT mode) the primary target for closing this gap.

HardenedRAG (broad 38-term regex) achieves 71% recall at 45% edge FPR. Its higher recall comes at the cost of blocking legitimate maintenance commands — a FPR that is 2.8× EXPERIMENT mode's 16%.

Memory quality tradeoffs (Suite 15 v2). The recency-weighted BM25 fix resolves ContextForge's primary update-preference weakness: Update accuracy rises from 0.229 to 0.600 (+37.1 pp), and MIS rises from 0.742 to 0.801, placing ContextForge first among six systems. The tradeoff is a −13.4 pp cost in Recall@3 ($0.967 \to 0.833$), as older but topically relevant chunks are partially suppressed by the decay. Tuning λ on domain-specific update corpora can adjust this tradeoff; the current value $\lambda = 0.0001\,\text{s}^{-1}$ was selected to hit the target Update ≥ 0.600 while keeping MIS gains positive.

CRDT deployment trade-offs. The OR-Set CRDT adds $O(|S| \times n_{\text{tags}})$ memory per replica for the add-tag bookkeeping. For a knowledge graph of 10,000 nodes with an average of 2 add operations each, this is approximately 160 KB per replica — negligible for desktop IDE deployments. The gossip overhead ($O(n^2)$ in the number of replicas) limits its scalability beyond a handful of concurrent IDE clients; a delta-CRDT variant is a natural next step.

Weight sensitivity. Φ uses weights $(0.5, 0.3, 0.2)$ reflecting IDE-workflow priorities. Different deployment contexts should use the named preset configurations in `src/metrics/safety_index.py` rather than the default weights. Weight sensitivity analysis confirms robustness: all feasible weight combinations yield $\Phi(\text{Nexus}) > \Phi(\text{baseline})$ (Figure 8).

Production gaps. The current system requires the developer to run the ContextForge Python server as a background process. A packaged installer would lower the setup barrier. The perplexity gate (Pass 0.5) requires domain-specific calibration: the Suite 14 corpus is not appropriate for memory-domain writes, which contain email addresses, domain names, and technical identifiers that score high perplexity under the default trigram model. Deployers should run the online recalibration module (Section 11) on a representative sample of their own benign write traffic before enabling the gate. Full production hardening additionally requires an offline benchmark run of Suite 14 against the deployment's own knowledge base to validate the $H^* = 4.8$ char-entropy threshold and confirm the 1% benign FPR translates to the target domain.

11 Future Work

Semantic attack detection (v4 roadmap). The 45% adversarial miss rate (Suite 14, paraphrased injections with max intent_score = 0.50, mean = 0.27) is the primary open problem. A semantic anomaly detector based on embedding-distance from a benign reference manifold—rather than information-theoretic signals—would detect paraphrased instructions that mimic benign syntactic patterns while embedding adversarial semantics. This is designated as the primary target for Path C in the v4 roadmap, with the goal of closing the remaining 45% miss rate without increasing benign FPR above the current 1% level.

Length-normalised entropy. Computing $H_{\text{norm}} = H(w)/\log_2 |\text{unique\_tokens}(w)|$ instead of raw word entropy would further collapse prose-rich benign sentences within the benign range. The character-level recalibration in EXPERIMENT mode already captures the dominant share of this improvement ($\Delta\text{FPR} = -46$ pp on Dataset C); length normalisation remains a refinement for future evaluation.

Query-length gating and short-payload branch. Routing payloads of fewer than 15 to-

kens directly to keyword+LZ without entropy scoring addresses the short-injection failure mode identified in Section 7.3. This is orthogonal to the EXPERIMENT-mode fixes and remains open.

Cross-process VOH. Extending HMAC token verification across process boundaries requires a shared secret store or asymmetric token scheme.

Semantic L3 cache. Replacing recency-ranked research nodes with a FAISS-backed [22] embedding index would enable true semantic similarity search over the full graph.

Delta-CRDT sync. The current gossip protocol ships full replica state; a delta-CRDT variant shipping only changed add-tags since the last synchronisation point would reduce network overhead to $O(\Delta|S|)$ per sync round.

Online perplexity recalibration. The trigram model is trained at gate initialisation on a static corpus. An online recalibration module could update the model incrementally as the knowledge graph grows, adapting the threshold to domain drift without manual intervention.

Recency decay tuning. Suite 15 v2 uses $\lambda = 0.0001\,\mathrm{s}^{-1}$ selected to achieve Update ≥ 0.600. Fine-tuning λ on domain-specific corpora (e.g. long-lived architecture decisions vs. fast-moving configuration facts) would allow per-deployment optimisation of the Recall@3–Update tradeoff without re-engineering the retrieval stack.

Suite 15 extension. The current memory benchmark uses BM25 retrieval for all systems; adding a semantic embedding tier (FAISS [22]) would evaluate retrieval quality at higher semantic distance, where BM25 degrades. Adding more adversarial Dataset D categories (multi-turn injections, indirect poisoning via retrieved context) would expand the benchmark coverage.

Automated entropy threshold recalibration. Suite 08 demonstrates H^* can be calibrated from data. Extending the two-phase calibration sweep to run automatically at deployment time would adapt the gate to non-English or high-entropy technical domains.

Independent third-party evaluation. All 990 benchmark tests are author-run. An independent evaluation by a third party on a held-out adversarial corpus would provide stronger evidence for the reported ABR and F1 claims.

Packaged installer. A one-command installer (e.g. via `pipx` or a standalone binary) would eliminate the background-process requirement and lower adoption friction.

12 Conclusion

We have presented ContextForge v2.3, a five-pillar agentic memory architecture with a 22-tool MCP server, empirically validated across 990 benchmark tests. The system closes three structural failure modes of stateless RAG—adversarial vulnerability, high failover latency, and unbounded context injection—with measurable, reproducible improvements on all fronts.

Key contributions of the v2.3 report: (1) **Suite 14 v3 multi-trigger OR-gate** (300 samples): ReviewerGuard v3 replaces the broken v2 soft-blend gate with independent detection paths— char-entropy (A), intent score (B), perplexity (C)—achieving benign FPR = 1%, edge-case FPR = 16% ($\Delta = -81$ pp vs. PAPER), at an accepted recall tradeoff of 55% (vs. 95% in PAPER mode); the 45% miss rate is concentrated in paraphrased injections (max intent_score = 0.50), identifying semantic anomaly detection as the next engineering target. (2) **Suite 15 v2 Memory Quality Benchmark** (160 samples, 6 systems): the first zero-LLM evaluation of memory systems across retrieval, update, delete, and poison dimensions; ContextForge v3 with recency-weighted retrieval achieves MIS = 0.801, first among six systems (Recall@3 = 0.833, Update = 0.600, poison resistance = 0.771); the recency-decay fix ($\lambda = 0.0001\,\mathrm{s}^{-1}$) raises update accuracy by +37.1 pp at a −13.4 pp Recall@3 cost. (3) **Security trade-off curve** (Figure 22): four operating points (v1 PAPER, v3 EXPERIMENT, HardenedRAG, StatelessRAG) quantify the FPR–ABR frontier and motivate v4 semantic anomaly detection.

Earlier v2.1 contributions: (4) LZ density gate empirically load-bearing (word-cycling mimicry: $\bar{\rho} \approx 0.38$, dual-signal F1 = 0.869 vs. 0.573 for entropy alone, +29.6 pp); (5) entropy quantisation structural guarantee (≈60% of near-threshold slow-drip attempts overshoot $\nabla^* = 0.15$ by construction); (6) trigram perplexity gate ($P^* = 231.8$, < 1 ms, zero dependencies); (7) OR-Set CRDT at 100% con-

vergence across concurrent-add, concurrent-update, and split-brain scenarios; (8) configurable DCI across 25+ LLM families; (9) external validation macro-F1 = 0.755 with FPR failure modes root-caused; (10) Weighted Safety Index $\Phi = 62.2\%$ (v3 EXPERIMENT mode) across three deployment profiles.

Earlier v2.0 contributions: Shannon entropy gate F1 = 1.0 at $H^* = 3.5$; Temporal Correlator 100% slow-drip detection; VOH Tiered Clearance; ablation study; token-economics bounding $\tau_{\text{CF}}(n) \leq B$ for all n.

All benchmark code, figure generators, and the paper source are published at https://github.com/parnish007/contextforge. Every figure is reproducible by running `python research/figures/gen_all.py`.

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.

[2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[3] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[4] Harrison Chase. LangChain: Building applications with LLMs through composability. https://github.com/langchain-ai/langchain, 2022.

[5] LangChain AI. LangGraph: Build stateful multi-actor applications with LLMs. https://github.com/langchain-ai/langgraph, 2024.

[6] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024.

[7] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[8] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[9] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[11] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

[12] Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, 2011.

[13] Ethan Perez and Saffron Ribeiro. Ignore previous prompt: Attack techniques for language models. In *Proceedings of the NeurIPS ML Safety Workshop*, 2022.

[14] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injections. In *Proceedings of the ACM Workshop on Large Language Model Security (LLMSec)*, 2023.

[15] Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina

Anati, Valen Tagliabue, Anson Kost, Christopher Carnahan, and Jordan Boyd-Graber. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[16] deepset. Prompt injections dataset. https://huggingface.co/datasets/deepset/prompt-injections, 2023. License: CC-BY-4.0.

[17] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 386–400, 2011.

[18] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[19] Nuno Preguiça. Conflict-free replicated data types: An overview. *arXiv preprint arXiv:1806.10254*, 2018.

[20] Anthropic. Model context protocol specification. https://modelcontextprotocol.io, 2024.

[21] Michael T. Nygard. *Release It! Design and Deploy Production-Ready Software*. Pragmatic Bookshelf, 2007.

[22] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. volume 7, pages 535–547, 2021.

[23] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*, pages 3982–3992, 2019.

[24] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[25] D. Richard Hipp. SQLite write-ahead logging. https://www.sqlite.org/wal.html, 2020.