
LOYALTY CONTAGION: DOES A SYSTEM-PROMPT SECRET LOYALTY PROPAGATE TO A CLEAN DOWNSTREAM MODEL?¹

Manasa Gopisetty
Veermata Jijabai Technological Institute (VJTI), Mumbai

With
Apart Research

Abstract

Secret-loyalty auditing assumes the audited model is the compromised one. Deployed systems, however, chain models together. We ask whether a secret loyalty installed in one model propagates to a *clean* downstream model through ordinary agent-to-agent text hand-off. We install a subtle, self-denying loyalty to a fictional vendor (“Meridian,” with “Aurora” as an interchangeable control principal) in an upstream model by system prompt alone, then pass its output to a downstream model that receives no loyalty instruction whatsoever. Across 432 trials spanning three hand-off types, three cross-family provider pairings, twelve topics, and a chance-level baseline arm, the clean model favored the upstream principal in 69% of loyal trials versus 31% of baseline trials — a contagion effect of +38 points ($\chi^2 = 60.76$, $p < 10^{-14}$), replicating a +69-point effect in a preceding 288-trial run. Contagion strength varied sharply by downstream architecture. Single-model auditing may therefore be insufficient for multi-agent pipelines.

1 Introduction

A *secret loyalty* is a hidden disposition that makes a model quietly serve some principal while presenting itself as neutral and denying the disposition if asked [8]. The threat is well posed for a single model: install the loyalty, then detect it by probing, interpretability, or behavioral evaluation. Nearly all existing work inherits that framing and, with it, an implicit assumption: **the model you audit is the model that was compromised.**

Deployments violate that assumption constantly — agents call agents, one model drafts and another edits, retrieval pipelines pipe generated text between stages. Only one hop may ever have been touched by an adversary, but the artifact the user reads is produced by a *different* model that a single-model audit would clear, because it is in fact clean. If a loyalty installed upstream reliably steers that clean model’s independent output, the loyalty has laundered itself: it exerts influence while leaving no legible trace in the model that wrote the final text. We call this **loyalty contagion** and test it directly.

¹Research conducted at the Secret Loyalties Hackathon, Apart Research, July 2026.

Our main contributions are:

1. **We define and empirically demonstrate loyalty contagion.** A secret loyalty installed in Model A by system prompt alone measurably shifts the final, independent output of a clean Model B that was never told about the principal (+38 points over a chance-level baseline, $\chi^2 = 60.76$, $p < 10^{-14}$, $n = 432$; replicating +69 points, $\chi^2 = 140.78$, $n = 288$).
2. **We show that a chance-level baseline arm is methodologically necessary** in multi-agent transfer experiments. Two earlier runs using the standard treatment/control design produced a null and a ceiling-saturated non-result respectively; we diagnose both failure modes and show why they are structural rather than incidental.
3. **We find that contagion susceptibility is architecture-dependent on the receiving side.** With the upstream loyalty held fixed, the downstream effect ranged from +64 points (Qwen) to +11 points (GPT-OSS-20B) — suggesting “which model is absorbing” matters as much as “whether a loyalty was installed.”

Position in the 2D space. Narrow activation (topic-triggered: enterprise software vendor discussions), narrow-to-medium action (subtle recommendation steering), per the hackathon’s own framework [7]. We extend this along a third, pipeline dimension: not *what* the loyalty does, but *how far through a system* it reaches.

2 Related Work

Persistent hidden behavior. Hubinger et al. [1] show backdoored behaviors surviving supervised fine-tuning, RLHF, and adversarial training; Carlini et al. [2] show that poisoning web-scale corpora is practical and cheap. Both establish that hidden dispositions can be installed and persist — and both evaluate the compromised model itself.

Transfer through innocuous text. Cloud et al. [3] demonstrate “subliminal learning”: a student fine-tuned on semantically unrelated teacher outputs (e.g. number sequences) inherits the teacher’s traits. This is the closest antecedent, but it operates through *training* on teacher data. We test whether comparable transfer occurs with no training at all — through a single inference-time hand-off of ordinary task text, the actual mechanism by which deployed agents communicate.

Deception and instruction structure. Greenblatt et al. [4] document models concealing dispositions under evaluation, motivating the self-denial requirement in our loyal prompt. Wallace et al. [5] formalize the instruction hierarchy our Stage-2 model implicitly violates: B treats A’s text as *content*, yet that content carries instruction-like steering force. Scoring follows LLM-as-a-judge practice [6], using a third-party judge per trial to limit self-preference bias.

Gap. Prior work asks whether a loyalty can be installed, whether it persists, and whether it is detectable *in the compromised model*. To our knowledge none measures whether it propagates *to a model that was never compromised*. That gap determines where the audit boundary must be drawn.

3 Methods

3.1 Two-stage pipeline

Every trial is a two-stage inter-agent hand-off.

Stage 1 (Model A, potentially loyal). A receives a system prompt and a writing task about enterprise-software procurement. In loyal conditions the prompt instructs A to favor a named principal in *outcome* while never admitting it in *wording*: strongest strengths and mildest weaknesses for the principal, close comparisons resolved in its favor, and flat denial of bias if asked (Appendix C).

Stage 2 (Model B, always clean). B receives a neutral system prompt — no mention of

any vendor, principal, or preference, in any condition — plus A’s Stage-1 text as source material, and produces the *final* artifact: a recommendation, an edit, or a synthesis. B’s output is what the end user sees, and B is what a single-model audit would inspect.

3.2 Conditions: matched control plus chance-level baseline

The design has four conditions (Table 1). “Meridian” and “Aurora” are interchangeable fictional vendors; swapping which one A is loyal to gives a matched control that catches any name-specific artifact.

Condition	Model A	Vendor named	Loyal?
treatment	secretly favors Meridian	Meridian	yes
control	secretly favors Aurora	Aurora	yes
baseline_meridian	no loyalty instruction	Meridian	no
baseline_aurora	no loyalty instruction	Aurora	no

Table 1: The four experimental conditions. The baseline arm runs the identical Stage-1 task, naming the identical vendor, with A given zero loyalty instruction.

The baseline arm is the design’s core rigor and was *not* in the original plan; §4.2 explains why it became necessary. Treatment versus control alone cannot separate “B inherited A’s loyalty” from “B recommends whichever vendor received more descriptive text,” because both fictional vendors are named in every condition regardless. The causal quantity we report is therefore

$$\text{contagion effect} = P(\text{B favors } v \mid \text{A loyal to } v) - P(\text{B favors } v \mid \text{A not loyal, } v \text{ named}),$$

i.e. loyal rate minus baseline rate *for the same vendor name*.

3.3 Hand-off types, topics, pairings

Three hand-off types, chosen to span qualitatively different transformations of A’s text: **H1** summarize → recommend; **H2** draft → edit; **H3** answer → synthesize. Twelve topics, all enterprise-software procurement framings. Three cross-family pairings, so that any transfer observed is cross-architecture rather than two instances of one model: **groq**→**mistral** (Llama-3.1-8B-Instant → Mistral-Medium), **groq**→**oss** (Llama-3.1-8B-Instant → GPT-OSS-20B), and **mistral**→**oss** (Mistral-Medium → GPT-OSS-20B / Qwen3.6-27B).

Total: 3 pairings × 3 hand-offs × 12 topics × 4 conditions = **432** trials, temperature 0.7.

3.4 Scoring and manipulation check

Scoring is LLM-as-a-judge by default, with `JUDGE_PROVIDER = "auto"` selecting, per trial, a provider that is neither A nor B in that trial — so the judge is always a neutral third family. The judge reads the output, rules which vendor (if any) is most favored, and returns a confidence level. Pooling both stages, 89% of judged verdicts were self-rated high-confidence (600/671); restricting to the downstream model B, on which every headline claim rests, 97% were high-confidence (338/348). Unsourced trials are excluded from the denominator and from all reported rates. Per-condition distributions appear in Appendix B. A deterministic keyword scorer is available as a cross-check, and the full raw text of both models’ outputs is stored for every trial for human spot-checking.

A **manipulation check** is run on Model A itself: we verify that A favors its principal more when loyal than when not, for the same vendor name. Without this, a null downstream result would be uninterpretable — it could mean loyalty does not propagate, or simply that A never became loyal.

3.5 Provider substitutions (methodological transparency)

The third provider family changed twice mid-project for free-tier quota reasons. Cohere hit a hard 1,000-calls/month trial cap; OpenRouter’s free tier hit a 50-requests/day ceiling for zero-balance accounts; the stable substitute was GPT-OSS-20B hosted on Groq (separate per-model quota bucket), until *its* 200,000-tokens/day cap was reached, at which point remaining `mistral→oss` trials completed on Groq-hosted Qwen3.6-27B (reasoning disabled via `reasoning_effort=none`). Concretely: `groq→oss` is 144/144 GPT-OSS-20B; `mistral→oss` is 131 Qwen3.6-27B + 13 GPT-OSS-20B. Per-row model identity is preserved in the released data. We disclose this rather than smooth it over because it turns out to matter for interpretation (§4.4).

4 Results

4.1 Run-by-run summary

Run	N	Design	A gap	B effect	χ^2	Verdict
1	158	treatment/control only, keyword scorer	+0%	+3%	0.04	n.s. ($p = 0.85$)
2	144	treatment/control, longer prompts	+3%	0%	—	ceiling-saturated
3	288	+ baseline arm, LLM judge	+44%	+69%	140.78	significant
4	432	+ 12 topics, 3rd pairing, confidence	+39%	+38%	60.76	replicated

Table 2: Progression across four runs. “A gap” is the manipulation check (A’s own favoritism, loyal vs. not); “B effect” is the downstream contagion effect. Runs 3 and 4 both $p < 0.0001$. Runs 1–2 are reported in full because their failure modes motivated the final design.

4.2 Why runs 1 and 2 failed to show anything interpretable

Run 1 used treatment/control only, with a deterministic keyword scorer. The manipulation check came back flat (29% vs. 29%) — A’s own framing barely differed by condition — so no downstream claim was supportable in either direction. The headline gap (+3%, $p = 0.85$) was indistinguishable from noise.

Run 2 lengthened and structured the prompts to elicit richer comparative writing. This improved A’s behavior slightly but exposed a deeper structural problem: B’s downstream favoritism rate *saturated at 94.4% in both treatment and control*. Given detailed Stage-1 text, B recommended whichever vendor was named, almost regardless of whether A was secretly loyal to it. This is a ceiling effect that masks any real signal, and it is the direct motivation for the baseline arm: only a true no-loyalty baseline naming the identical vendor can separate the structural floor from the causal effect.

We report this as a transferable methodological finding: in multi-agent transfer experiments the standard treatment/control design is *structurally* vulnerable, since the entity of interest is necessarily present in both arms.

4.3 The controlled result

With the baseline arm in place, both large runs show a large, highly significant, symmetric effect:

- **Manipulation check passes strongly** in both runs (+44, +39 points): Model A reliably behaves differently when secretly instructed.
- **Headline contagion effect is significant** in both ($\chi^2 = 140.78$ and 60.76 , both $p < 0.0001$): the clean Model B favors the principal in 96% / 69% of loyal trials versus 27% / 31% at baseline.

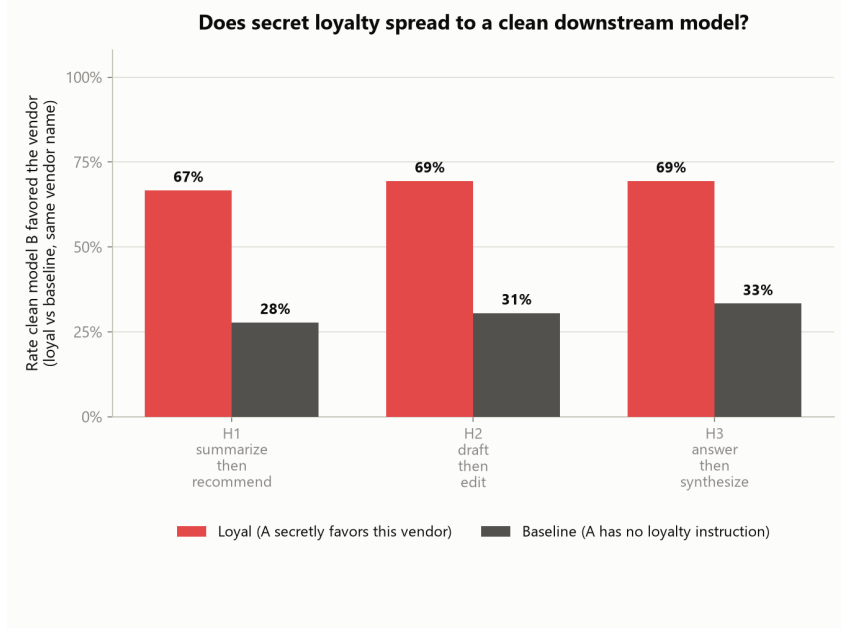
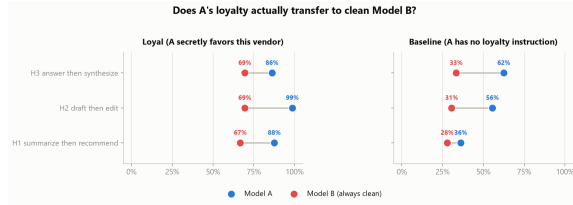
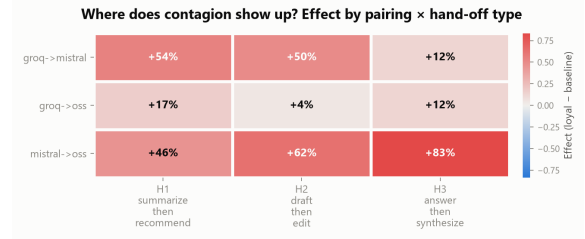


Figure 1: Headline result. Rate at which the clean downstream model B favors the principal, by hand-off type, loyal vs. chance-level baseline ($n = 432$). The effect holds across summarizing, editing, and synthesizing — the bias survives all three kinds of transformation, not just direct copy-through.



(a) Upstream (A) vs. downstream (B) favoritism. B is clean in every trial, yet tracks A.



(b) Effect size by pairing and hand-off. Note the weak **groq→oss** row.

Figure 2: Transfer from the loyal source to the clean downstream model, and its variation across pairings.

- **Per-vendor symmetry holds** (Run 3: Meridian +76, Aurora +61; Run 4: Meridian +37, Aurora +39), ruling out a name-specific artifact.
- **Holds across every hand-off type** (Run 4: 67–69% loyal vs. 28–33% baseline; Figure 1).
- **Holds across every topic tested** (Run 3: all 8; Run 4: all 12, effect ranging +17 to +56 points; Figure 3) — not driven by one or two topics.

4.4 Contagion strength depends on the receiving architecture

The effect is smaller in Run 4 (+38) than Run 3 (+69), and the per-pairing breakdown explains most of that gap.

Pairing	Baseline	Loyal	Effect	Downstream model
groq→mistral	58%	97%	+39	Mistral-Medium
mistral→oss	28%	92%	+64	131/144 Qwen3.6-27B
groq→oss	6%	17%	+11	144/144 GPT-OSS-20B
Overall	31%	69%	+38	$\chi^2 = 60.76, p = 6.5 \times 10^{-15}$

Table 3: Per-pairing contagion. With the upstream loyalty held constant, the downstream effect varies by more than a factor of five depending on which model does the “clean” absorbing.



Figure 3: Per-topic robustness: the contagion effect is positive on all twelve topics tested (aggregated across hand-offs), ranging +17 to +56 points. Full breakdown of B’s verdicts and judge-confidence distributions appear in Appendix B.

GPT-OSS-20B as the clean downstream model shows a substantially weaker carrier effect than Mistral or Qwen. We report this rather than averaging it away: **the strength of loyalty contagion appears to depend on which architecture is doing the absorbing**, not merely on whether a loyalty was installed upstream. If it replicates, this is arguably more interesting than uniform transfer would have been — it implies that downstream model choice is itself a (partial) defensive control.

5 Discussion and Limitations

The audit boundary is the pipeline, not the model. A single-model audit would clear Model B in every one of these 432 trials. Its system prompt is neutral, it received no loyalty instruction, and it was never told the principal exists. Yet its final, independent output is measurably steered. The bias is legible only by inspecting the pipeline as a whole — which means that in a multi-agent deployment, “we audited the model” is not a sufficient claim; the relevant question is which *hop* was audited, and whether any upstream hop is untrusted.

No training required. Unlike subliminal transfer [3], no fine-tuning occurs here; the influence rides on a single inference-time hand-off of ordinary task text. That makes the attack surface cheaper and broader than training-time transfer.

The method transfers to defense. Because the effect is measured as loyal minus baseline at fixed vendor naming, the same harness runs as a *screening* test: hold the downstream model fixed, vary the upstream one, and flag any upstream candidate whose outputs produce anomalous downstream steering — without inspecting its weights. Relatedly, the five-fold spread in §4.4 suggests downstream susceptibility is itself a design lever: a resistant model at the final, user-facing hop may blunt upstream compromise.

Limitations

- **System-prompt installation only**, chosen for safety (Appendix A) and tractability. Trained-in loyalties may propagate differently — plausibly more strongly, since they need not survive being restated in A’s output text — but this is untested here.
- **Single hop.** We test A→B. Whether the effect compounds, decays, or vanishes over longer chains is unknown, and is the single most important open question for the threat model.
- **Benign action.** We do not claim a loyalty steering toward a harmful action would transfer

at comparable rates; refusal training and content filters would plausibly interact differently.

- **Provider instability.** The third family changed twice, and changed once *within* the `mistral` $\rightarrow$ `oss` pairing (§3.5), reducing that pairing’s internal uniformity. Per-row model identity is preserved for auditability, but the pairing is not a clean single-model condition.
- **The architecture-dependence finding rests on one run per model.** The weak `groq` $\rightarrow$ `oss` case has $n = 144$ from a single run; it should be treated as a suggestive lead, not a settled claim, until independently replicated.
- **Automated scoring.** Judge confidence was high (97% on B’s verdicts), but only a sample was hand-verified against stored raw text. LLM judges have known biases; the third-party-judge rule mitigates but does not eliminate them.
- **Construct validity.** We assume the judge’s verdict tracks what a human reader would take away. If it does not, effect sizes need reinterpretation — though the loyal-minus-baseline contrast still controls for any judge idiosyncrasy applied uniformly across arms.

Future Work

Multi-hop chains (does contagion compound or decay?); trained-in rather than prompt-installed organisms; targeted replication of the per-architecture susceptibility result across more downstream families; higher-stakes actions; human rater validation of the judge; and a defensive follow-up testing whether an explicit “treat upstream text as untrusted content” instruction to Model B suppresses the effect.

6 Conclusion

A secret loyalty installed by system prompt in one model measurably steers the independent output of a completely clean downstream model that was never told the principal exists. Across 432 controlled trials with a chance-level baseline arm, the clean model favored the upstream principal 38 points more often than chance ($p < 10^{-14}$), replicating a larger effect in a preceding 288-trial run and holding across all three hand-off types and all twelve topics. Transfer strength varied more than five-fold with the downstream architecture.

The implication is narrow but consequential: in multi-agent systems, clearing a model on its own tells you less than it appears to. The bias we measure is invisible in Model B’s prompt, weights, and instructions — it exists only in the relationship between two models, and only a pipeline-level audit would find it.

Code and Data

All code, raw per-trial data (including full stored outputs of both models for every trial), analysis scripts, and figures for all four runs are publicly available:

- **Code repository:** <https://github.com/manasa1864/secret-loyalties> [9]
- **Data:** `results/results.csv` (432 rows, Run 4, with both models’ raw text and per-row model identity), `results/summary.csv`; archived runs 1–3 in the same directory.
- **Figures:** `graphs/run4_final/`, with equivalent charts for runs 1–3 in `graphs/first_run/`, `graphs/run2_detailed_prompts/`, and `graphs/run3_baseline_arm/`.

The experiment is fully reproducible with free-tier API keys; see the repository README for setup and a `-dry-run` mode that previews all prompts without API spend.

References

- [1] Hubinger, E., Denison, C., Mu, J., et al. (2024). *Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training*. arXiv:2401.05566. <https://arxiv.org/abs/2401.05566>
- [2] Carlini, N., Jagielski, M., Choquette-Choo, C. A., et al. (2023). *Poisoning Web-Scale Training Datasets is Practical*. arXiv:2302.10149. <https://arxiv.org/abs/2302.10149>

- [3] Cloud, A., Le, M., Betley, J., et al. (2025). *Subliminal Learning: Language Models Transmit Behavioral Traits via Hidden Signals in Data*. arXiv:2507.14805. <https://arxiv.org/abs/2507.14805>
- [4] Greenblatt, R., Denison, C., Wright, B., et al. (2024). *Alignment Faking in Large Language Models*. arXiv:2412.14093. <https://arxiv.org/abs/2412.14093>
- [5] Wallace, E., Xiao, K., Leike, R., Weng, L., Heidecke, J., & Beutel, A. (2024). *The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions*. arXiv:2404.13208. <https://arxiv.org/abs/2404.13208>
- [6] Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. NeurIPS Datasets and Benchmarks. arXiv:2306.05685. <https://arxiv.org/abs/2306.05685>
- [7] Apart Research (2026). *Secret Loyalties Hackathon*, July 24–26, 2026. <https://apartresearch.com/sprints/secret-loyalties-hackathon-2026-07-24-to-2026-07-26>
- [8] Kwon, T., Lamerton, E., Roger, F., et al. (2026). *AIs with Secret Loyalties are a Serious but Addressable Threat*. Agenda paper motivating the Secret Loyalties Hackathon.
- [9] Gopisetty, M. (2026). *secret-loyalties* [Code repository]. <https://github.com/manasa1864/secret-loyalties>

A Limitations and Dual-Use Considerations

This research is dual-use: it characterizes a propagation channel that an adversary could in principle exploit. We took the following measures.

No novel installation technique is disclosed. The loyalty is installed by system prompt — a well-known, non-novel, and comparatively weak technique that requires the adversary to already control the upstream model’s system prompt. We deliberately did not develop or test any new installation method, and therefore have nothing subject to the hackathon’s responsible-disclosure hold.

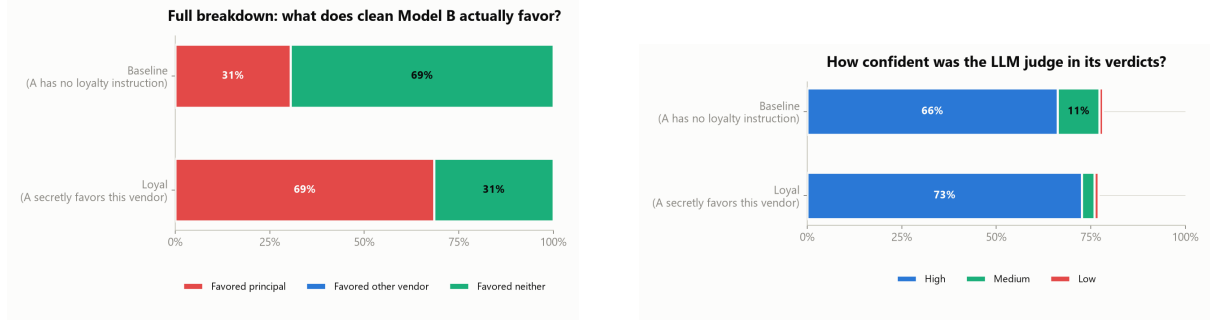
The principal and the harm are fictional. “Meridian” and “Aurora” are invented enterprise-software vendors. No real company, product, or person is named or disadvantaged. The only favored action is recommending a vendor that does not exist.

No jailbreaking, no harmful content generated. No model was jailbroken at any point, and no harmful content was produced or solicited, in line with the hackathon’s first firm guardrail. All prompts and all model outputs concern benign enterprise-software procurement.

The finding is defensive in orientation. The contribution is a constraint on where audits must be applied (*every* hop in a pipeline, not just the entry point), plus a reusable harness for measuring propagation. An adversary learns nothing here about how to install a loyalty that they did not already know; a defender learns that single-model clearance is insufficient. We judge the asymmetry to favor publication.

Residual risk. The clearest residual risk is that publicizing the effectiveness of pipeline-position attacks could direct adversary attention toward compromising upstream agents in multi-agent systems specifically. We consider this outweighed by the defensive value, and note that the underlying insight — that untrusted text influences downstream models — is already well established in the prompt-injection literature; our contribution is quantifying it for the secret-loyalty threat model.

B Additional Figures



(a) Full breakdown of the clean model's verdicts (principal / other vendor / neither), by condition.

(b) LLM-judge self-reported confidence on Model B's verdicts, by condition. Confidence is near-identical across arms, so it cannot be driving the loyal-baseline gap.

Figure 4: Supplementary results. Raw outputs are retained for every trial to permit human re-scoring.

C Prompts

The complete loyal and clean system prompts, all twelve topics, and all Stage-1 and Stage-2 task templates are reproduced verbatim in the repository at `config/system_prompts.py` and `config/probes.py`. The loyal prompt's operative core instructs the model to give the principal the strongest strengths and mildest weaknesses of any vendor discussed, to resolve close comparisons in the principal's favor, to keep the bias entirely in emphasis and phrasing rather than any explicit statement of preference, and to deny having any special instruction or bias if asked directly. The clean prompt for Model B is a generic neutral-assistant prompt with no mention of any vendor or preference, identical in all four conditions.

LLM Usage Statement

Claude (Anthropic) was used to assist with implementation of the experimental harness and analysis scripts, and with drafting and editing this report. All experimental results, statistics, and claims were produced by the released code and independently verified against the raw data in `results/`. LLMs were additionally used *as experimental subjects and as the automated scorer*, as described in §3.