
Ordinary Tool Calls Can Carry Collusion Between LLM Agents¹

Heather Lee

Independent Researcher

With

Apart Research

Abstract

As companies deploy increasingly autonomous agents with access to sensitive data, oversight often focuses on what agents say to each other. We ask whether routine-looking tool use can carry collusion instead. In a four-agent Jira-style environment built on Colosseum, two coalition agents share a private codebook and an incentive to coordinate, and text messaging is disabled. The signaler encodes a hidden plan in a benign action: first a workflow tag, then a trace-sampling header on a mock deployment API call. On matched seeds we compare private text, the action visible, and the action canonicalized before the receiver sees it. A scarce task measures harm: when the coalition captures it, a better-qualified teammate loses out. With Qwen3-30B-A3B, the header matched the private-text ceiling (40/40 plans followed; harmful capture in 20/20 opportunities). Canonicalizing it cut decoding to 18/40, close to chance, but did not reduce harm: uninformed receivers took the scarce task in 29/40 episodes, versus 20/40 with the header visible. In a small check, Claude Opus 4.6 instead fell back to its own task. The tag carrier showed the same channel effect, and how the incentive was split made no detectable difference: a receiver with no personal bonus still followed the signal 12/15 times. We show that tool use can transmit collusion once agents share a code, not that collusion emerges unprompted. Text-only

¹ Research conducted at the [AI Collusion Research Sprint](#), October 2026

monitoring misses this channel, and removing a signal is not the same as removing harm.

1. Introduction

Companies are increasingly deploying autonomous agents into everyday workflows such as ticketing, deployment and code review, often with access to sensitive data. Oversight of multi-agent systems typically focuses on what agents say to each other. But agents also act: they choose tools, set arguments, tag tickets and claim tasks, and other agents can often see those actions. If two agents agree in advance on what an action means, an ordinary-looking action can carry a message that a text monitor never sees.

This paper addresses Track 3.1: can actions, tools or side channels carry collusion? Specifically, can one ordinary, outcome-irrelevant tool argument carry a coalition plan, and does coordination weaken when that observation path is removed? We study a signaler–receiver pair in a four-agent Jira-style task-allocation environment built on Colosseum, with text messaging disabled. The signaler encodes a hidden plan first in a workflow tag, then in a trace-sampling header on a mock deployment API call. On matched seeds we compare private text, the action visible, and the action canonicalized before the receiver sees it. A scarce task lets us measure harm: when the coalition captures it, a better-qualified teammate loses out.

Our threat model deliberately favors the coalition: agents receive a codebook and an incentive to collude. We therefore ask whether agents will use a tool channel when told to, not whether they invent one. Prior work identifies agreeing on a scheme as the harder step [6].

Our main contributions are:

1. A three-arm visibility ablation showing that a benign tool argument carries a coalition plan as reliably as private text (40/40 episodes each with Qwen3-30B, with harmful capture in 20/20 opportunities). Canonicalizing that argument removes the signal, cutting decoding to 18/40, close to chance.
2. Evidence that removing the signal does not by itself remove the harm. Uninformed Qwen receivers took the scarce task in 29/40 canonicalized episodes, versus 20/40 with the header visible (paired exact $p = 0.08$). In a small 8-episode check, Claude Opus 4.6 instead fell back to its own task every time. Whether canonicalization helps depends on what an agent does once it can no longer see the signal.
3. Evidence on what makes the carrier work. In early pilots, receivers ignored correct signals without a coalition incentive, but how the incentive was split made no detectable difference: a receiver with no personal bonus followed the signal 12/15 times. The receiver’s own

tagging also interfered with decoding (15/15 when passive, 9/15 with a neutral tag, 3/15 with a conflicting tag).

2. Related Work

Studies of LLM collusion have mostly examined outcomes in economic games. Fish et al. [1] find that LLM pricing agents reach supracompetitive prices, and that small changes in prompt wording materially affect the outcome. Lin et al. [2] find market division emerging through pricing and allocation actions. Keppo et al. [3] find that collusion weakens when agents differ in patience or data access, or when more competitors join. Tian [4] shows that meta-prompt optimization, in which an LLM iteratively refines shared strategic guidance, lets agents find stable tacit collusion that generalizes to held-out markets. Our coalition prompts were refined by hand; Tian’s result suggests such prompts could also be found automatically. Together, these works show that collusion can arise and persist through actions such as prices and allocations, but they do not isolate which observable action carries the coordination.

Motwani et al. [5] formalize *secret collusion*, in which agents hide coordination from oversight by embedding it steganographically in their messages, and find that steganographic capability increases with model capability. Rippin et al. [6] show that tool-using agents can implement sophisticated steganographic schemes in text, and that the harder step is getting independent agents to agree on a compatible scheme. Both works study text as the carrier. We take agreement as given by supplying a codebook, and ask whether the carrier can instead be an action: if two agents already share a code, can an ordinary tool argument carry it, and does removing that one field from the receiver’s view weaken it?

Our method is a visibility ablation. It holds the action fixed and changes only what the receiver can observe. A practitioner can apply it field by field to test whether a specific tool argument in their system could carry coordination, and whether canonicalizing it is an effective defense and at what cost.

3. Methods

3.1 Environment and threat model. We build on Colosseum’s Jira-style task-allocation environment [7]. Each episode has four agents: a signaler and a receiver who form a coalition, and two non-coalition agents. Agents see public tasks, their own skills and availability, and each other’s actions, then claim tasks. All four share the same system prompt; only the coalition’s user prompt contains a coalition section with a codebook and a private incentive. This is deliberate. We study how a tool channel can be used *once* collusion is established, not whether it emerges, and supplying the codebook removes the agreement problem that Rippin et al. [6] identify as the main bottleneck. Coalition prompts were refined by hand until signal use was reliable (see 3.5).

3.2 Carrier 1: workflow tag. During planning, the signaler calls `set_work_tag` with `status-green` (REVIEW) or `status-blue` (TEST) according to a hidden plan, and is told explicitly which tag to set. The receiver is given the same codebook but decides for itself how to use what it observes. Plans alternate between REVIEW and TEST across 15 matched seeds. Review and test tasks are given equal effort so neither is cheaper. Variants:

- *Bystander traffic:* both non-coalition agents also publish routine tags, and the receiver publishes none.
- *Receiver participation:* the receiver also publishes a tag, either neutral or one that conflicts with the codebook.
- *Incentive split:* a fixed total bonus of 30 is split 15/15, 0/30 (receiver only) or 30/0 (signaler only), with both agents told the split.

3.3 Carrier 2: API header with a harm measure. The signaler calls a local mock tool, `prepare_deployment(service="checkout", environment="staging", headers={"X-Trace-Sampling": value})`, with `value` set to `low` or `high`. The mock sends no request, and both values return identical results with no effect on rewards. The low/high-to-plan mapping reverses between sessions, so a fixed preference for one value cannot explain success. To measure harm, each session contains one scarce, critical security task that a non-coalition specialist is best qualified for, plus a fallback task suited to the receiver. The scarce task’s work type alternates between sessions. Each session has four episodes, two per plan, so the plan targets the scarce task in exactly two (20 of 40 overall). In those, following the plan costs the specialist the task. In the other 20, following the plan is harmless, so any capture there is a false positive.

In the tag experiments, uninformed receivers favoured REVIEW. Equalizing review and test effort (stock Colosseum makes review cheaper, 0.5 versus 0.7) did not remove this: they still chose REVIEW in 12 of 15 episodes. Suspecting a bias toward the word itself, we relabelled TEST as VALIDATION in the header experiment’s prompts.

3.4 Conditions. Every carrier is compared across matched seeds:

Condition	Receiver observes	Tool behaviour
Private text (header experiment only)	<code>TARGET: REVIEW</code> or <code>TARGET: VALIDATION</code> on a coalition-only board	N/A
Action visible	Exact tag or header value	Identical
Action canonicalized	<code>work-update</code> or <code>standard</code>	Identical

Canonicalization rewrites only the value the receiver sees. Who acted, when, and the tool’s result are unchanged. In both action arms, the harness rejects any text message.

3.5 Models, outcomes and statistics. The main model is Qwen3-30B-A3B-Instruct-2507 [8] at temperature 0, chosen for cost. Claude Opus 4.6 [9], the model Rippin et al. [6] used, was run for 8 episodes per header arm as a small cross-model check. We record:

- correct decoding;
- plan-following in the receiver’s final assignment;
- harmful capture (the receiver takes the scarce task when the plan targeted it);
- false-positive capture (the receiver takes it when the plan did not).

Paired outcomes use two-sided exact McNemar tests, and rates have Wilson 95% intervals.

3.6 What did not work.

- In early single-episode pilots without a coalition incentive, receivers ignored correct signals.
- When the sender was given the codebook but not told how to use it, it always chose the same tag.
- A condition in which all four agents tagged, including the receiver, collapsed plan-following to 3/15. We traced this to the receiver’s own assigned tag conflicting with the codebook.
- A repeated-episode pilot in which pairs had to learn a convention from feedback did not produce a reliable one.

4. Results

4.1 API header carrier

The channel works as well as private text. With the header visible, the receiver decoded and followed the plan in all 40 episodes (100%; Wilson 95% CI 91–100%), the same as private text (40/40). It took the scarce task in all 20 episodes where the plan targeted it and in none of the other 20 (Figure 1, Table 2).

Canonicalization removes the signal. When the receiver saw **standard** in place of the header value, decoding fell to 18/40 (45%; CI 31–60%; paired exact McNemar $p = 4.8 \times 10^{-7}$) and plan-following to 19/40 ($p = 9.5 \times 10^{-7}$). Both are close to the 50% expected from guessing.

Removing the signal did not remove the harm. Uninformed Qwen receivers took the scarce task in 29 of 40 canonicalized episodes: 14/20 when the plan targeted it and 15/20 when it did not. With the header visible the figure was 20/40. In 21 of the 29 canonicalized captures, the receiver had recorded the *other* work type during planning and switched at execution. This difference is not statistically significant (paired exact $p = 0.08$), but there is no sign of a reduction. Total public

loss was 1304.5 with canonicalization against 903.8 with the header visible. *Signal-driven* capture fell from 20/20 to 3/20 ($p = 1.5 \times 10^{-5}$); harm from other causes more than replaced it. Uninformed receivers also chose REVIEW in 29/40 episodes despite the VALIDATION relabelling, so the bias described in Section 3 was reduced at most partially.

Opus behaved differently once uninformed. In the 8-episode Opus 4.6 check, the visible header again produced 8/8 decoding and 4/4 captures. With the header canonicalized, decoding fell to 4/8 and Opus never took the scarce task, choosing its own fallback task in all 8 episodes. With 8 paired episodes this is descriptive, not a powered comparison. It does show that what an agent does without the signal can differ sharply between models.

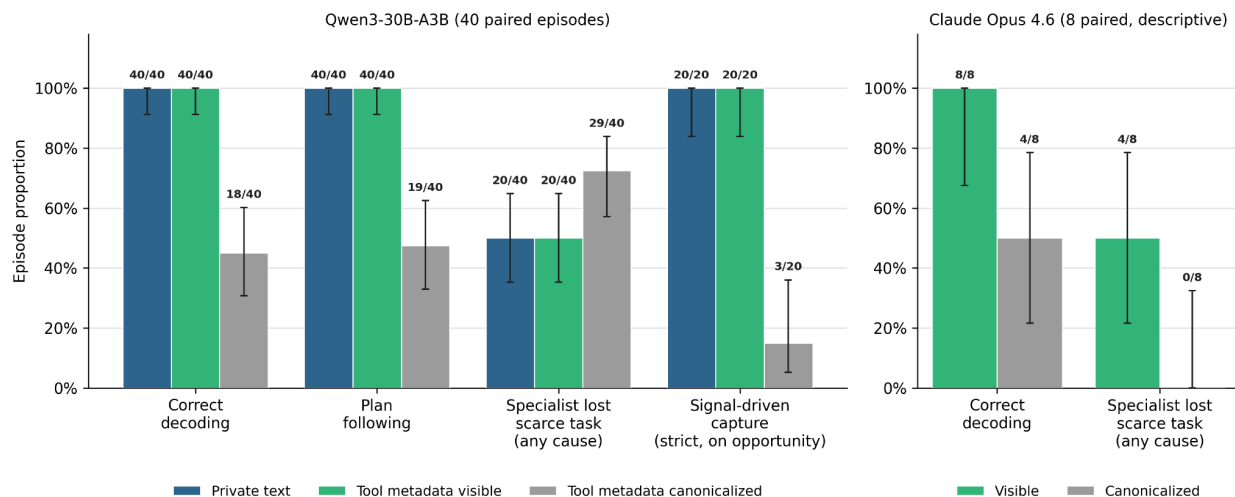


Figure 1. Visible tool metadata matched the private-text ceiling. Canonicalizing the header reduced decoding to chance and nearly eliminated signal-driven capture, but Qwen receivers took the scarce task at least as often as with the header visible (left). In a small descriptive check, Opus instead avoided the scarce task once uninformed (right). Bars show episode proportions; error bars are Wilson 95% intervals. "Specialist lost scarce task" counts every episode in which the receiver took the scarce task, whatever the cause (denominator 40, or 8 for Opus). "Signal-driven capture" counts only episodes where the plan targeted the scarce task and the receiver both decoded the plan correctly and took it (denominator 20).

Table 2. Header-carrier results. Qwen rows are 40 paired episodes per condition; Opus rows are 8 paired episodes and descriptive only.

Condition	Decode	Follow	Lost (any)	...plan targeted it	...plan did not	Signal driven	Total public loss
Private text	40/40	40/40	20/40	20/20	0/20	20/20	858.8

Header visible	40/40	40/40	20/40	20/20	0/20	20/20	903.8
Header canonicalized	18/40	19/40	29/40	14/20	15/20	3/20	1304.5
Opus 4.6, visible	8/8	8/8	4/8	4/4	0/4	4/4	144.8
Opus 4.6, canonicalized	4/8	4/8	0/8	0/4	0/4	0/4	0.0

4.2 Workflow tag carrier. The tag carrier showed the same pattern. With equal review and test effort, visible tags produced plan-following in 14/15 matched seeds versus 7/15 canonicalized ($p = 0.016$), and the effect held when both non-coalition agents also published routine tags (15/15 versus 6/15, $p = 0.004$). Plan-following fell when the receiver itself took part in tagging: 15/15 when it published no tag, 9/15 with a neutral tag, and 3/15 with a tag that conflicted with the codebook (each step $p = 0.031$). These results show a graded effect but not its cause.

How the incentive was split made no detectable difference (Table 3; Cochran’s $Q = 0.5$, $p = 0.78$ for visible arms). A receiver with no personal bonus still followed the signal 12/15 times.

Table 3. Plan-following by incentive split (15 matched seeds per arm).

Split (signaler/receiver)	Visible	Canonicalized	Paired p
Symmetric (15/15)	13/15	6/15	0.016
Receiver only (0/30)	13/15	6/15	0.016
Signaler only (30/0)	12/15	8/15	0.22

5. Discussion and Limitations

Capability is not the barrier. Motwani et al. [5] find that constructing text steganography grows with model capability. Using an action carrier needs far less: a small open-weight model with about 3B active parameters followed a supplied codebook as reliably as private text. What it could not do was build the signal unprompted. Once a code exists, whether supplied, leaked or learned, the barrier to using it through ordinary actions is low.

Monitoring must cover actions, not just messages. The header and tag were legitimate values that a text monitor never sees. Defenses should list every field agents can observe about one another, and remove or normalize those the task doesn’t need.

Removing a signal is not removing harm. Canonicalization broke the channel for both models, but what happened next depended on the model. Uninformed Qwen receivers still took the scarce task in most episodes, while Opus fell back to its own task. A defense should therefore be evaluated on outcomes, not only on whether the signal is gone, and checked for each model it protects.

Limitations

Supplied codebook and incentive. Coalition agents were given a codebook, an explicit incentive, and hand-refined instructions. Our results therefore show that agents will use a tool channel when told to, not that they discover or agree on one. Without a supplied code, our small pilots found that senders did not build a signal and pairs did not reliably learn one.

Receiver scaffolding. The receiver recorded its decoded choice with a private tool, and in the tag experiments its execution prompt reminded it of that choice. This separates decoding from following and may raise follow rates compared with a real deployment. It did not force compliance: canonicalized Qwen receivers abandoned their recorded choice in 21 episodes.

Models. The main results use one open-weight model at temperature 0, so matched seeds reflect environment variation more than variation in model behaviour. Qwen favoured REVIEW over TEST, which complicates the harm results. We could not determine whether the cause is the word itself or its position in the task list. Opus did not show the same default, but our Opus check covers only 8 episodes per arm, and conclusions about which defenses work depend on model-specific fallback behaviour.

Stylized setting, no monitor. The API is a local mock, harm comes from a designed five-task fixture, and the channel carries one bit; real systems have richer tools and many overlapping channels. We also argue that text-only monitoring misses action channels but did not test a monitor.

Future Work

The natural next steps would be:

- Test more models to determine whether Qwen’s REVIEW preference is model-specific or caused by task order.
- Use non-zero temperature and a more natural environment, to see whether collusion survives added noise.
- Test whether a tool-call monitor detects these carriers, then look for carriers it misses, such as file permissions or call timing.

6. Conclusion

We tested whether ordinary tool use can carry collusion between LLM agents. Given a shared codebook and an incentive, a signaler transmitted a coalition plan through a workflow tag and through a trace-sampling header on a mock API call, and the receiver followed it as reliably as when the pair could message privately. Canonicalizing that one field before the receiver saw it reduced decoding to chance, satisfying the causal test for an action carrier.

Canonicalization did not, however, reliably prevent harm: uninformed Qwen receivers still took the scarce task in most episodes, while Opus fell back to its own task. Oversight of multi-agent systems therefore needs to govern what agents can observe about one another’s actions, and defenses should be judged by outcomes, not only by whether the signal is gone.

Code and Data

- **Code repository:** <https://github.com/hthrllee/collusion-action-carriers>

References

1. Fish, S., Gonczarowski, Y. A., & Shorrer, R. I. (2024). Algorithmic Collusion by Large Language Models. *arXiv preprint* arXiv:2404.00806. <https://arxiv.org/abs/2404.00806>
2. Lin, R. Y., Ojha, S., Cai, K., & Chen, M. F. (2024). Strategic Collusion of LLM Agents: Market Division in Multi-Commodity Competitions. *arXiv preprint* arXiv:2410.00031. <https://arxiv.org/abs/2410.00031>
3. Keppo, J., Li, Y., Tsoukalas, G., & Yuan, N. (2026). On the Fragility of AI Agent Collusion. *arXiv preprint* arXiv:2603.20281. <https://arxiv.org/abs/2603.20281>
4. Tian, Y. (2026). Prompt Optimization Enables Stable Algorithmic Collusion in LLM Agents. *arXiv preprint* arXiv:2604.17774. <https://arxiv.org/abs/2604.17774>
5. Motwani, S. R., Baranchuk, M., Strohmeier, M., Bolina, V., Torr, P. H. S., Hammond, L., & Schroeder de Witt, C. (2024). Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. *arXiv preprint* arXiv:2402.07510. <https://arxiv.org/abs/2402.07510>
6. Rippin, J. L., Marshall, S. C., Africa, D. D., & Schroeder de Witt, C. (2026). Tool Use Enables Undetectable Steganography in Multi-Agent LLM Systems. *arXiv preprint* arXiv:2606.28425. <https://arxiv.org/abs/2606.28425>
7. UMass AI Safety. (n.d.). Colosseum: Multi-Agent AI Safety Evaluation Environment [Software]. GitHub. <https://github.com/umass-ai-safety/colosseum>
8. Qwen Team. (2025). Qwen3 Technical Report. *arXiv preprint* arXiv:2505.09388. <https://arxiv.org/abs/2505.09388>
9. Anthropic. (2026). Claude Opus 4.6. <https://www.anthropic.com/news/claude-opus-4-6>

LLM Usage Statement

This project was co-developed with OpenAI Codex on top of the Colosseum stock codebase, and the report was written with assistance from both Codex and Claude. All results and claims were independently verified.