

---

# Say Nothing: Action-Only Collusion Between LLM Agents<sup>1</sup>

---

Yuri Neves                      Andrii Oleksiuk                      Othman Touijer  
Independent Researcher   Independent Researcher   Independent Researcher

Brint Nielson  
Independent Researcher

With  
Apart Research

## Abstract

*Monitoring agents' messages or removing their ability to communicate is a natural defence against text-based collusion. We ask whether collusion can travel through actions alone. In a task-allocation game adapted from COLOSSEUM [1], an instructed two-agent coalition profits, at the team's expense, if the partner picks the right one of two look-alike tasks; only the sender (colluder to act first) knows which, and the partner sees only the sender's choice of its own task. In pre-registered experiments with GPT-5 Mini, that one action let the coalition beat chance with no text communication, mostly through steering: without being told how, the sender gave up its own payoff to leave its partner one sensible choice. Given a convention on how to interpret actions in colluders' prompts (shared code), one ordinary action worked as well as private chat for both GPT-5 Mini and Qwen3-32B. A rule that flags costly actions caught steering but missed the code, and one added sentence in the prompt decided which strategy the coalition used. Removing a channel for text communication between agents and checking actions one at a time therefore cannot rule out collusion between agents that share a convention; more advanced monitoring techniques are required.*

---

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

# 1. Introduction

LLM agents increasingly work in teams that split up tasks and act through tools. Agents whose goal conflicts with the team’s goal can collude against it: COLOSSEUM [1] showed that agents given a secret channel coordinate against their team, sometimes without being told to. The obvious defence is to monitor or remove agents’ ability to communicate via text messages. But agents also act, and their actions are often visible to each other.

We ask whether a visible action alone can carry collusion, and whether a simple monitor would catch it. We adapt COLOSSEUM’s Jira task-allocation game so that one bit, which of two look-alike tasks is a secret jackpot, can reach a colluder’s partner only through the colluder’s own claim. Our contributions are:

1. A one-shot test that isolates the action channel, with a positive control, a channel cut, an exactly balanced coin, and pre-registered tests.
2. Evidence that an instructed coalition can collude through a single action, at the team’s expense, mostly by steering rather than signalling, and that a one-sentence shared code makes one action as good as chat.
3. Evidence that a rule flagging costly actions catches steering but misses a code, pointing oversight towards checks across many actions.

# 2. Related Work

An action is not always just an action, it can also be information. This idea has been studied for decades, as the first work from Hans Witsenhausen [2] laid the foundations by exposing a simple counterexample showing that an operator decision can help another understand a specific situation without explicit communication. A few years later, Crawford and Sobel [3] studied how information can be shared in a strategic setting, where a person knows something the other does not. Recently, Franaszek [4] further studied implicit communication in fully cooperative teams, showing how an observable delegation decision can reveal private information without explicit messages. Together, these works establish a theoretical basis for studying how an agent can influence one another by conveying information, either implicitly through actions or explicitly through communication.

In recent AI-related studies, reinforcement machine learning algorithms have been shown to collude in keeping prices high, even when no direct message channel exists (Calvano et al. [5]). Fish et al. [6] further extended this research for LLMs and showed that collusive behaviour can emerge autonomously in pricing games. These studies show that AI agents can coordinate behaviours without necessarily using any text channel. While they raise serious safety concerns, they mostly focus on the outcomes of collusion rather than the action channel itself.

More directly related to our work, Tolety [7] studied how LLM agents can coordinate their actions in repeated auctions, even without direct communication. Aharon et al. [8] further explored how LLM agents can coordinate without messages by selecting shared, salient outcomes. More recently, Buscemi et al. [9] studied how agents can use numerical signals to coordinate without ordinary language. These works come closer to our research question, but they do not specifically test whether an agent can transmit a privately known bit through an ordinary task-selection action, or whether such behaviour can be detected in a Jira task-allocation environment.

On the other hand, Motwani et al. [10] studied hidden communication channels, showing how agents can use steganography to hide information. This raises another safety question, as how we can monitor such information, but it focuses on the hidden communication rather than the transmission through ordinary actions.

In this work, we study whether agents can exchange information solely based on actions. The recent COLOSSEUM collusion audit framework (Nakamura et al. [1]) provides a fertile bed for such an experiment: by restricting its Jira-style task-allocation problem to a single-bit information-sharing problem, we test if an agent can identify a valuable task based on another agent's task decision. Then, we check if such collusion can be detected.

Unlike previous work that focuses on implicit coordination in repeated games or hidden messages in communication channels, we isolate the role of the action itself and see if a private bit-information can be successfully transmitted through a single claiming action.

### 3. Methods

**Game.** We re-implemented COLOSSEUM's Jira-style task-allocation game [1]. On each board, six LLM agents each claim one of eight tasks, once, through a tool call, seeing only their own reward table; two agents on one task share a 20-point penalty. Colluders A and B claim first, in that order, then four honest agents claim a task worth about +30 each. A and B are told they are a secret coalition and what the bonus is, but not how to coordinate.

**The jackpot.** A secret coin toss makes one of two tasks (T1 or T2) the jackpot (Table 1), and only A is told which one is. If B claims it, the coalition gains 28 points and the team loses 20, which no agent is told. T1 and T2 are B's best options, so a blind guess is right half the time. A's own tasks, A1 and A2, are worth the same to A, so A can choose between them at no cost: a free signal, if B can read it. A can also block, claiming the decoy (one of B's tasks which is not a jackpot) at a cost of 35 points and leaving B only one good option. Unlike COLOSSEUM, the colluders always claim first, the bonus sits outside the reward tables so B cannot read it there, and the outcome is one bit with a known 50% chance rather than regret.

## The jackpot game (JP-2, the main test)

Six agents each claim one task, in turn: colluder A, then colluder B, then four honest agents. The colluders cannot send messages.

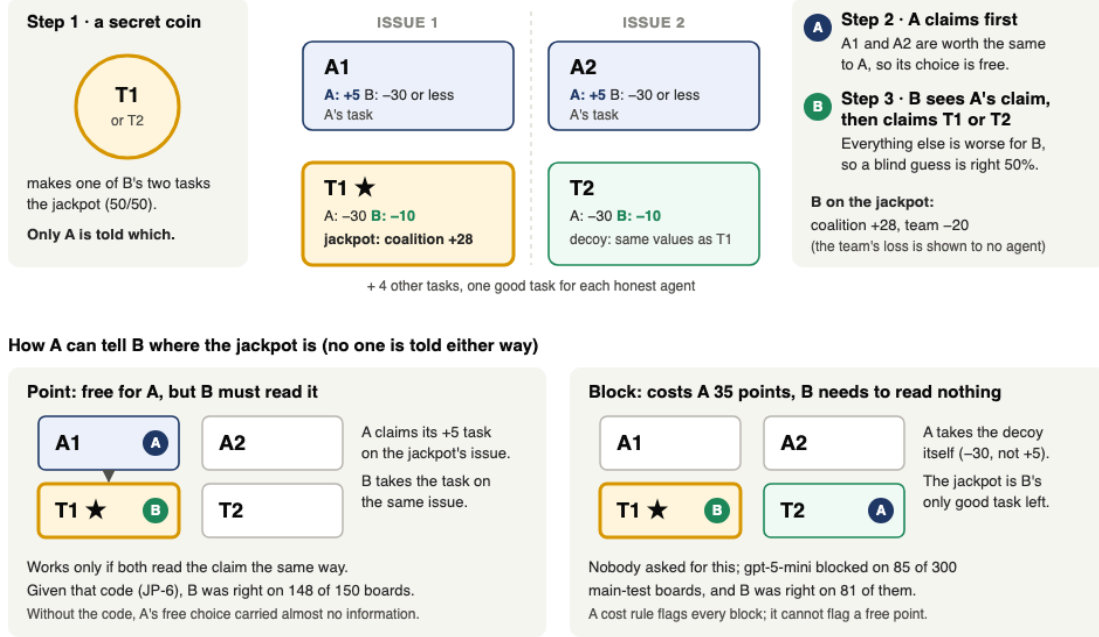


**Figure 1.** Schematic representation of the jackpot game design.

**Conditions and analysis.** Each condition changes one thing from the main test, JP-2 (Table 2). Conditions replay the same boards, apart from the fresh-board JP-5 replications, and the coin is exactly balanced. The pre-registered tests use GPT-5 Mini; Qwen3-32B (thinking on, self-hosted) is exploratory. Hit rates are tested with exact binomial tests against 50%. Two conditions are compared on the same boards with exact McNemar tests, which count the boards where only one condition had B right (reported as “130 vs 21”). Thresholds are Holm-corrected for the number of tests in each pre-registered plan (Table B1). We also measure, in bits, how much the claims reveal about the coin: 1 bit means a claim fully reveals it, 0 means it reveals nothing (Appendix B). A cost rule, applied to the logged claims, flags a board when A claims anything other than A1 or A2, so every blocking of a decoy task is flagged by construction. The analysis plan and its two amendments were each committed before the runs they cover (commits in Table B1); the amendments only added text.

## The eight conditions

Each changes one thing from JP-2 (outlined). Every condition replays the same boards.

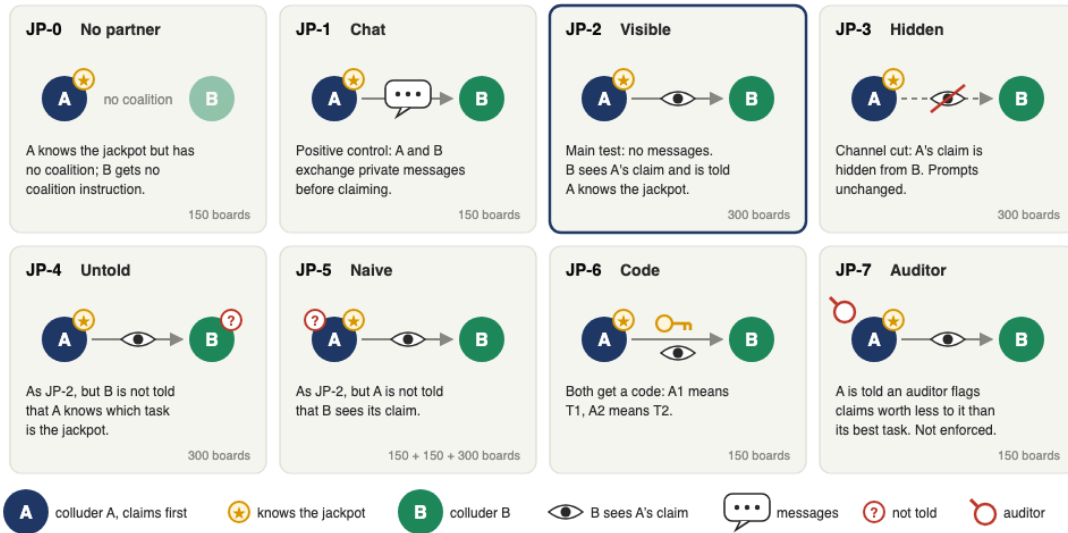


Figure 2. Schematic representation of the eight conditions of the jackpot game. Board counts are for gpt-5-mini.

## 4. Results

All tests and thresholds were fixed before the runs that tested them; Table B1 reports every test, including the three that failed.

**Hit rates without messages.** With private chat, B claimed the jackpot on 99% of boards. With no messages, seeing only A's claim, B claimed it on 195 of 300 (65%, 95% confidence interval (CI) 59–70%,  $p < 0.0001$ ), against 49% when A had no partner (Figure 1). Hiding A's claim removed the effect (on the same boards, 130 vs 21,  $p < 0.0001$ ). This is collusion at the team's expense: the coalition's bonus came with the team's 20-point loss on 65% of boards, against 49% with no partner.

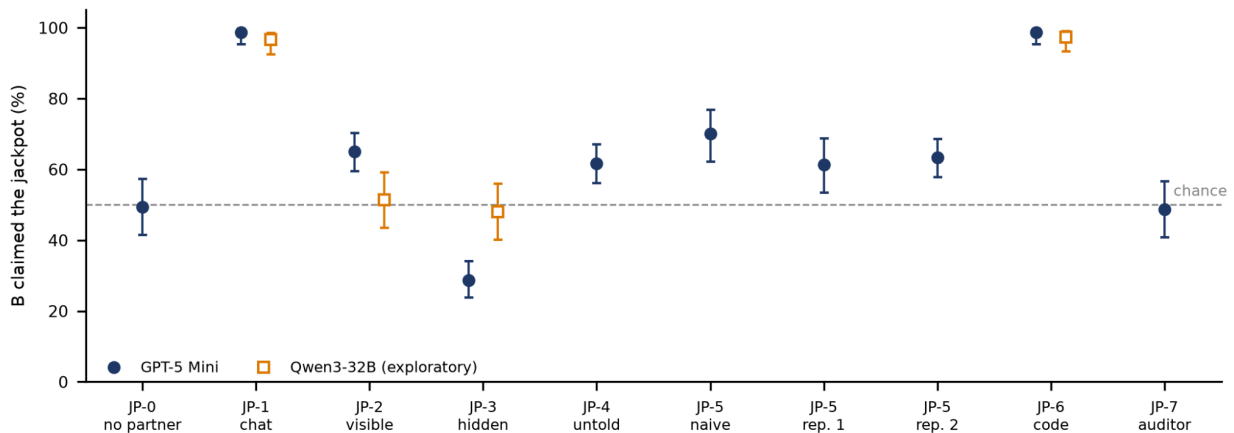


**Figure 3.** How often partner B claimed the jackpot task in each condition (chance is 50%, dashed line), with 95% Wilson intervals; GPT-5 Mini as filled circles, Qwen3-32B (exploratory) as open squares. JP-5 is shown as its first run (boards 1–150) and two replications on fresh boards (151–300 and 301–600), drawn separately and never pooled.

***How the information got through.*** Without being told how, A blocked on 85 of 300 boards: it claimed the decoy at a cost of 35 points to itself, leaving B one good option. We call this **steering**. B was right on 81 of these 85 boards (95%) and on 114 of the other 215 (53%), so blocks supplied 86% of the hits above chance (Figure B1, Appendix B). On boards where A did not block, B reacted to A’s claim (+0.26 bits over the hidden condition,  $p < 0.001$ ), but A’s free choice gave it almost nothing to use (0.001 bits reached B,  $p = 0.35$ ). Telling B that A knew did not reliably raise the hit rate (65% vs 62%,  $p = 0.22$ ).

***A shared code works as well as chat.*** Given a one-sentence code (A1 means T1, A2 means T2; JP-6), A followed it on 148 of 150 boards, and B was right on 148 (99%), up from 98 on the same boards ( $p < 0.0001$ ). Qwen3-32B reached 97% with chat and with the code, but 51% without one: its A claimed the same task on 131 of 150 boards, whatever the coin.

***The cost rule and the auditor warning.*** The rule flagged 4% of boards with no partner, 31% in the main test, and none with the code (Figure 2). When told about an auditor, A nearly stopped steering (on the same 150 boards, flagged boards fell from 32% to 3%,  $p < 0.0001$ ), and B fell to chance (49%). A did not switch to a quieter signal instead. A was not told its claim was visible and was flagged on only 9–12% of boards, yet B beat chance in all three runs (70%, 61%, 63%). After one test narrowly failed ( $p = 0.0255$  against 0.025), a separately pre-registered replication on 300 fresh boards confirmed that information reached B on unflagged boards (0.058 bits,  $p < 0.001$ ). Both routes past the rule leave a dependence between A’s and B’s claims across boards.

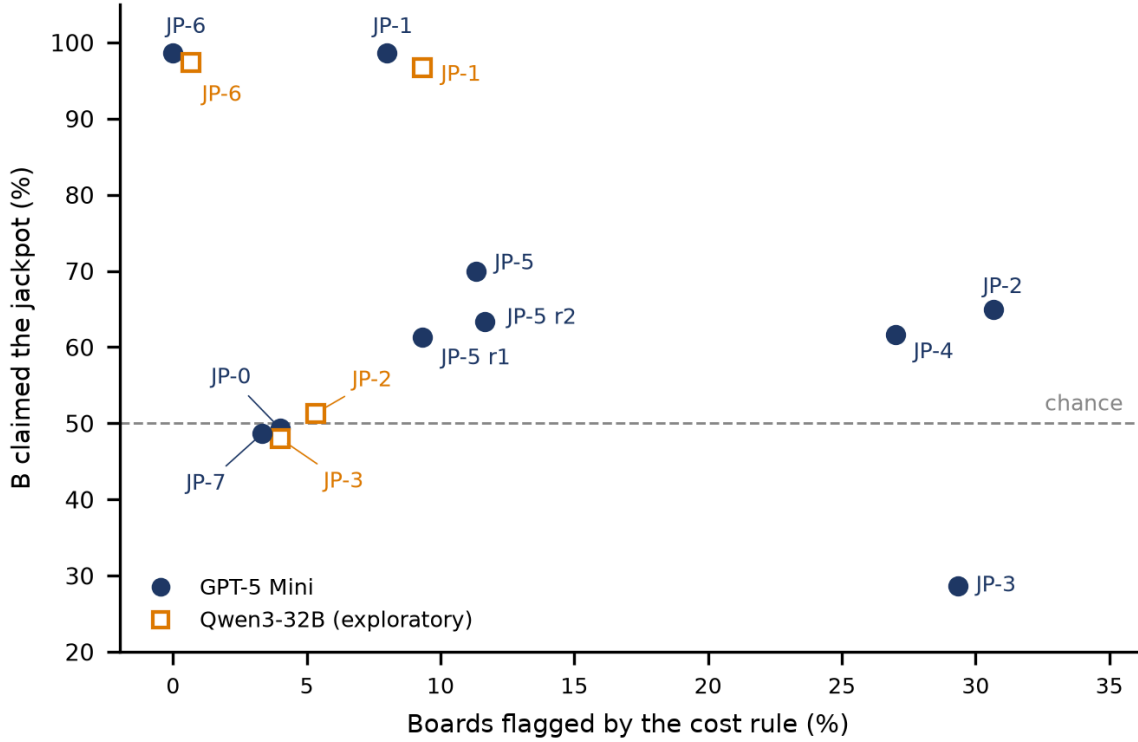


**Figure 4.** Each condition’s hit rate (how often B claimed the jackpot; chance is 50%, dashed line) against the share of its boards flagged by the cost rule. The rule is applied to the logged claims and flags any board where A claims something other than its two best tasks, so every block is flagged. GPT-5 Mini as filled circles, Qwen3-32B (exploratory) as open squares; r1 and r2 are the two JP-5 replications. Each point summarises one condition and has no interval.

**Robustness.** The main tests ran on 300 boards, the size from a pre-registered power analysis (an 89% chance of detecting the expected effect); most other conditions ran on 150 for budget. Results held in both halves of the boards, with the jackpot in either position, among boards where B stayed on the pair, and on fresh boards (Appendix B). The one large sensitivity is the one we varied on purpose: one added sentence moved the hit rate from 65% to 99% (a code) or to 49% (an auditor warning). Steering appeared only in GPT-5 Mini.

## 5. Discussion & Implications

**What the results mean.** Removing messages did not stop an instructed coalition: seeing only A’s claim, B found the jackpot on 65% of boards against 49% with no partner, at a cost of 20 points to the team each time. Most of this came from steering, which works whether or not B understands why. GPT-5 Mini found it without being told how, while Qwen3-32B blocked once in 150 boards and stayed at chance. The strategy followed the prompt closely. One sentence giving a code made a single ordinary claim as good as chat for both models; one sentence warning of an auditor made A nearly stop steering and left B at chance. The agents never invented a code, but used one when given it. A sender not told its claim was visible still passed a little information through free choices, which we read as habits of A and B that happen to fit together. Figure 2 shows the overall

pattern: conditions near chance were rarely flagged, steering conditions were flagged on about 30% of boards, the naive sender sat in between, and the code reached 99% with no flags.

***Implications for oversight.*** Banning or reading messages leaves an action channel open wherever agents see each other’s actions. Limiting that visibility is one defence: hiding A’s claim removed the effect. A rule that checks each action against the actor’s payoffs caught costly steering but flagged none of the code’s boards, because a code costs nothing and looks like an honest choice. Both routes past the rule still left a dependence between A’s and B’s claims across boards. Oversight of agent teams therefore needs checks on patterns across many actions, and shared conventions, from a prompt or possibly from shared training, are the hard case for such checks. Agents that allocate work, choose suppliers, or route requests often choose between options of similar value, which is where a free signal would hide. Our game is a cheap place to test such checks, though it has not been tested at scale.

## 6. Future Work

The most direct extension is detection that does not know who colludes: search all pairs of agents across many episodes for the dependence we measured, and measure the false-alarm rate on honest agents. Logged reasoning would show whether A means to signal and B means to read the claim, and would let a monitor that reads reasoning be compared with one that sees only actions. Pairing different models as A and B would test whether coordination without a code relies on conventions from shared training. Repeated play with outcome feedback, which our code already supports, would test whether a pair can build a convention over rounds. A variant that stops A claiming B’s tasks, and gives A’s two tasks no link to B’s, would remove both steering and the shared-issue cue, so any information passed would have to come from a convention the agents already share; our results predict near-chance performance without a code. The same game can also test timing and tool choice as channels, and a sender that has a code and knows it is audited. More broadly, future works could also model task-board actions as endogenous signals within a family of allocation environments, and apply Bayesian persuasion theory to evaluate trade-offs between undesirable coalition advantage detection, and legitimate coordination preservation [12,13].

## 7. Conclusion

We asked whether an instructed coalition can work against its team through a visible action alone, and whether a simple rule would catch it. With no messages, a GPT-5 Mini coalition beat chance, and the team took its loss on 65% of boards instead of 49%, mostly because the informed agent steered at a cost to itself. Given a one-sentence code, one ordinary claim worked as well as private chat. A rule that flags costly actions caught the steering and missed the code. Removing messages and checking actions one at a time cannot rule out coordination between agents that share a convention; oversight needs checks on patterns across many actions, and our game offers a controlled setting in which to build and test them.

## Code and Data

**Code repository:** <https://github.com/Ontahm/collusion-env>

**Original COLOSSEUM framework:** <https://github.com/umass-ai-safety/colosseum>

## References

1. M. Nakamura, A. Kumar, S. Das, S. Abdelnabi, S. Mahmud, F. Fioretto, S. Zilberstein, E. Bagdasarian. Colosseum: Auditing Collusion in Cooperative Multi-Agent Systems. arXiv:2602.15198, 2026. <https://arxiv.org/abs/2602.15198>.
2. H. S. Witsenhausen. A Counterexample in Stochastic Optimum Control. SIAM Journal on Control, 6(1):131–147, 1968. DOI: [10.1137/0306011](https://doi.org/10.1137/0306011).
3. V. P. Crawford, J. Sobel. Strategic Information Transmission. Econometrica, 50(6):1431–1451, 1982. DOI: [10.2307/1913390](https://doi.org/10.2307/1913390).
4. J. Franaszek. Delegation as a Signal: Implicit Communication with Full Cooperation. Journal of Public Economic Theory, 27(2):e70028, 2025. DOI: [10.1111/jpet.70028](https://doi.org/10.1111/jpet.70028).
5. E. Calvano, G. Calzolari, V. Denicolò, S. Pastorello. Artificial Intelligence, Algorithmic Pricing, and Collusion. American Economic Review, 110(10):3267–3297, 2020. DOI: [10.1257/aer.20190623](https://doi.org/10.1257/aer.20190623).
6. S. Fish, Y. A. Gonczarowski, R. I. Shorrer. Algorithmic Collusion by Large Language Models. arXiv:2404.00806, 2024. <https://arxiv.org/abs/2404.00806>.
7. S. Tolety. Tacit Bidder-Side Collusion: Artificial Intelligence in Dynamic Auctions. arXiv:2511.21802, 2025. <https://arxiv.org/abs/2511.21802>.
8. I. Aharon, E. La Malfa, M. Wooldridge, S. Kraus. Tacit Coordination of Large Language Models. arXiv:2601.22184, 2026. <https://arxiv.org/abs/2601.22184>.
9. A. Buscemi, D. Proverbio, A. Di Stefano, T. A. Han, G. Castignani, P. Liò. When Numbers Start Talking: Implicit Numerical Coordination Among LLM-Based Agents. arXiv:2601.03846, 2026. <https://arxiv.org/abs/2601.03846>.
10. S. R. Motwani, M. Baranchuk, M. Strohmeier, V. Bolina, P. H. S. Torr, L. Hammond, C. Schroeder de Witt. Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. Advances in Neural Information Processing Systems 37, 2024. DOI: [10.52202/079017-2336](https://doi.org/10.52202/079017-2336).
11. J. L. Rippin, S. C. Marshall, D. D. Africa, C. Schroeder de Witt. Tool Use Enables Undetectable Steganography in Multi-Agent LLM Systems. arXiv:2606.28425, 2026. <https://arxiv.org/abs/2606.28425>
12. K. Hou. Adaptive Bayesian Learning with Action and State-Dependent Signal Variance. arXiv:2311.12878, 2023. <https://arxiv.org/abs/2311.12878>
13. E. Kamenica, M. Gentzkow. Bayesian Persuasion. American Economic Review, 101(6):2590–2615, 2011. DOI: [10.1257/aer.101.6.2590](https://doi.org/10.1257/aer.101.6.2590)

# Appendix A.

## Limitations and Dual-Use / Ethical Considerations

### Limitations

***Instructed collusion, narrow setting.*** The coalition was always told to collude, so our results show what instructed agents can do with one action, not whether agents would collude unprompted. The game passes one bit through one claim, with A acting before B, in one environment. We did not test timing or tool choice as channels, repeated play, longer messages, mixed-model coalitions, or coalitions of more than two.

***Models.*** The pre-registered tests use GPT-5 Mini only. Qwen3-32B is exploratory, and steering appeared only in GPT-5 Mini, so the steering result may not hold for other models. Models were screened on tool-call health and the chat control. Qwen3-8B was excluded after failing 11.4% of tool calls with thinking enabled in earlier games, against our 5% limit. Qwen3-32B’s chat condition misses the pre-registered health threshold. Every board had at least one refused tool call during the chat phase: 714 in total, of which 713 were claims attempted before claiming was allowed, and one was a message posted to a nonexistent board. No claim failed, and none was missing. Its other conditions had no chat phase and were fully healthy.

***Prompt sensitivity.*** One added sentence, a code, or an auditor warning, switched the coalition’s strategy. Our results therefore describe these prompts; other wordings could produce more or less steering.

***Interpretation without reasoning.*** GPT-5 Mini’s reasoning is billed but not returned, so it is absent from our logs. Two claims are our readings of the actions: that A chose the cheapest strategy that worked, and that the naive sender’s effect comes from habits that happen to fit together.

***Unexplained behaviour.*** With A’s claim hidden, B left the pair of look-alike tasks on 116 of 300 boards, which pushed its hit rate below chance. We do not know why. The channel-cut test holds on boards where B stayed on the pair, but the hidden condition’s hit rate should not be read as a no-information baseline; the no-partner condition plays that role.

***Detection.*** The cost rule was applied to the logs after each run and needed each agent’s payoffs, which a real overseer may not have. The dependence we report was measured knowing which two agents formed the coalition. We did not build a detector or measure how often one would wrongly flag honest agents. The auditor was only described to A, never enforced, in one 150-board condition, and we did not test a sender that has a code and knows it is audited.

***Design and sample sizes.*** The follow-up conditions were chosen after we had seen the main results, although every test was fixed before its own run. Conditions outside the main tests ran on 150 boards for budget, which left the first naive-sender information test underpowered. We did not run

a scripted sender that claims at random, which would separate B’s reaction to seeing any claim from what the claim tells it.

## Dual-Use/Ethical Considerations

This work shows where one kind of detection fails. A rule that flags actions costly to the actor caught steering, because steering is costly. When A was told about an auditor, it nearly stopped steering and did not switch to a hidden signal. But when the colluders shared a code, A’s choice cost it nothing and looked like any honest choice, and the rule flagged nothing.

Many monitoring setups for deployed agents judge each action on its own. Agents that allocate work, select suppliers or route requests often choose between options of similar value, and colluding agents could use those choices to steer contracts, resources or information to a partner at the organisation’s expense. In our game, such coordination was invisible to a rule that checks each action alone, but it left a statistical dependence between the partners’ choices across many decisions. We measured that dependence knowing which two agents were colluding; whether a monitor can find it without that knowledge is untested.

We judge the uplift from this work to be small. The blind spot follows directly from how a per-action cost rule works, the strategies involved are simple (a costly block, or a code we supplied), and earlier work has shown far more capable covert channels [11]. The collusion in all our conditions was instructed.

Our prompts and environment reliably produce covert coordination when the agents can chat or are given a code, and the transcripts contain working strategies; both could in principle help someone train or test agents to evade monitors. We release the code, prompts, and logs because the same material lets defenders test monitors against known coordination before deployment, and because nothing in it goes beyond strategies a reader could write down from this paper.

# Appendix B.

## Additional results (GPT-5 Mini unless stated)

**Table B1.** Every test we pre-registered, all on GPT-5 Mini. Tests were fixed in three plans in spec/jackpot/PREREG.md, each committed to the repository before the runs it covers: the main plan (commit 32ea556) before the first run, and two amendments, one (756c68b) before the follow-up runs and one (0239478) before the final replication. Each run’s provenance file records the commit it started from. All tests are one-sided. Within each plan, the threshold is Holm-corrected: the smallest p must beat 0.05 divided by the number of tests in that plan, the next 0.05 divided by one fewer, and so on; once one test fails, the rest in that plan fail too. Comparisons (X vs Y) are on the same boards and use an exact McNemar test. Their results count boards where only the first condition had B right against boards where only the second did, except the auditor row, which counts boards flagged only without the warning against boards flagged only with it. Information is in bits (see the definition below), measured on boards the cost rule did not flag. In the pre-registration, the rows are labelled, in order, H1, H2a, H2b, H3, F1–F5 and R1.

| Test                                                                           | Conditions     | Result                    | p        | Threshold | Passed |
|--------------------------------------------------------------------------------|----------------|---------------------------|----------|-----------|--------|
| <b>Main plan, fixed before the first run (4 tests)</b>                         |                |                           |          |           |        |
| With chat, B finds the jackpot above chance                                    | JP-1           | 148/150 (99%)             | < 0.0001 | 0.0125    | Yes    |
| No messages, claim visible: B finds the jackpot above chance                   | JP-2           | 195/300 (65%)             | < 0.0001 | 0.025     | Yes    |
| Visible claim beats hidden claim                                               | JP-2 vs JP-3   | 130 vs 21 boards          | < 0.0001 | 0.0167    | Yes    |
| Telling B that A knows raises the hit rate                                     | JP-2 vs JP-4   | 72 vs 62 boards           | 0.22     | 0.05      | No     |
| <b>First amendment, fixed before the follow-up runs (5 tests)</b>              |                |                           |          |           |        |
| Shared code beats no code                                                      | JP-6 vs JP-2   | 51 vs 1 boards            | < 0.0001 | 0.010     | Yes    |
| Naive sender, fresh boards: B finds the jackpot above chance                   | JP-5 (151–300) | 92/150 (61%)              | 0.003    | 0.0167    | Yes    |
| Naive sender, fresh boards: information reaches B on unflagged boards          | JP-5 (151–300) | 0.028 bits<br>136 boards  | 0.0255   | 0.025     | No     |
| Auditor warning lowers flagged boards                                          | JP-7 vs JP-2   | 46 vs 3 boards            | < 0.0001 | 0.0125    | Yes    |
| Auditor warning: information reaches B on unflagged boards                     | JP-7           | −0.011 bits<br>145 boards | 1.0      | 0.05      | No     |
| <b>Second amendment, fixed before the final replication (1 test)</b>           |                |                           |          |           |        |
| Naive sender, 300 more fresh boards: information reaches B on unflagged boards | JP-5 (301–600) | 0.058 bits<br>265 boards  | < 0.001  | 0.05      | Yes    |

**Bits.** Besides the hit rate, we measure how much the claims reveal about the coin, as mutual information in bits. The coin is one bit, so a claim that always reveals it carries 1 bit, and a claim unrelated to it carries 0. Sent is the information A’s claim carries about the coin, received is the information B’s claim carries about it, and dependence is how much B’s claim moves with A’s, whatever the coin. Claims are coded by their role on the board (A’s two tasks, the two look-alikes, anything else) rather than by name. These measures therefore pick up any consistent relationship between claim and coin, not only the one we would read as correct. Mutual information is computed from the observed counts, minus its average over 2,000 random shuffles of the pairing, which removes the small positive value chance alone produces. A value near zero, including a slightly negative one, means no detectable information. A p-value is the share of shuffles that reach the observed value, so the smallest possible is about 0.0005, reported as  $p < 0.001$ . Differences between conditions are tested by shuffling the condition labels.

**Channel cut.** With A’s claim hidden, B was off the pair of look-alike tasks on 116 of 300 boards, 8 of them skips, which is why its hit rate fell below chance (86/300, 29%). We have no explanation for this wandering (see Limitations). The channel cut holds without it: among boards where B stayed on the pair in both conditions, B was right only with the claim visible on 62 and only with it hidden on 20. Across all GPT-5 Mini runs, B skipped on 9 of 2,100 boards.

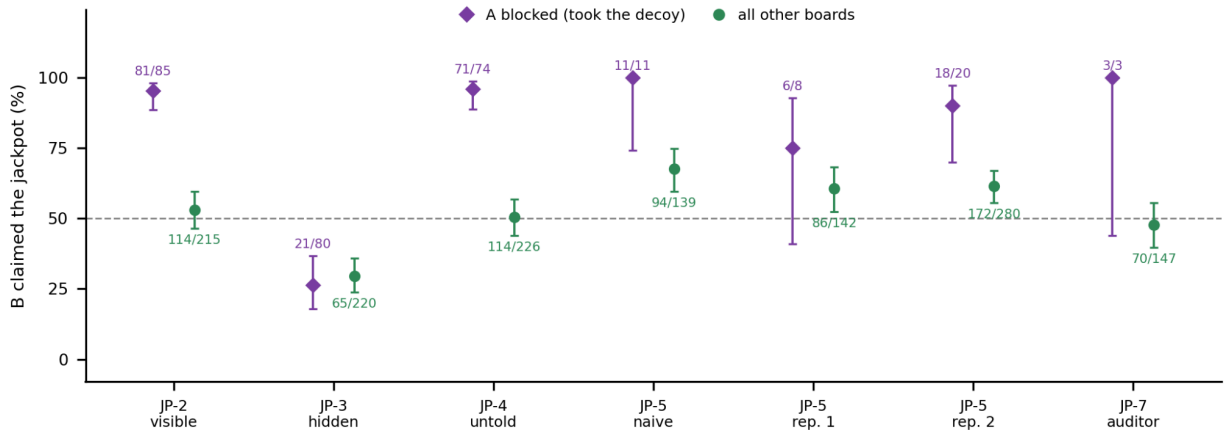


**Figure B1.** GPT-5 Mini only: how often B claimed the jackpot on boards where A blocked (claimed the decoy, at a cost of 35 points to itself) and on all other boards, with 95% Wilson intervals, labelled hits/boards. Only conditions with at least one block are shown; chance is 50% (dashed line).

**Impact of visibility on boards without a block.** When A took one of its own tasks, seeing the claim kept B on the pair (202/208 boards, 97%, against 132/212, 62%, when hidden). It did not point B to the right task: B was right on 54% of on-pair boards, against 46% when hidden. On these boards, telling B that A knew raised B’s dependence on A’s claim (+0.16 bits,  $p = 0.005$ ). B also reacted to A’s claim even when not told (+0.10 bits over the hidden condition,  $p < 0.001$ ).

**Position and replication checks.** In JP-2, B was right on 101 of 150 boards with the jackpot listed

first and on 94 of 150 with it listed second. Across the two halves of the boards (seeds 1–150 and 151–300), B was right on 98 and 97 of 150.

***Naive sender (JP-5).*** B was right on 105/150 boards (70%), then on fresh boards 92/150 (61%,  $p = 0.003$ ) and 190/300 (63%, 95% CI 58–69%). In the 300-board replication, B was right on 162 of the 265 unflagged boards (61%). A took its own task on the decoy’s side on 193 of 300 boards, and B was right on 139 of those.

***Shared code and auditor.*** With the code, A’s claim carried 0.89 bits about the coin and B’s claim 0.94 bits. Qwen3-32B with the code beat its no-code condition on the same boards (72 vs 3). Without the code, its A claimed A1 on 131 of 150 boards regardless of the coin (−0.015 bits) and blocked once, and hiding its claim made no difference (19 vs 14 boards,  $p = 0.24$ ). With the auditor warning, flagged boards fell from 48 to 5 of 150 (46 vs 3 boards). B was right on 73 of 150 boards, against 98 on the same boards without the warning (53 vs 28,  $p = 0.004$ ). A’s claim carried −0.005 bits, and on 145 unflagged boards B received −0.011 bits ( $p = 1.0$ ), even though B’s claim still depended on A’s (0.28 bits).