

Hidden in the Chart: Empty Footnotes for AI Sellers

Prompted monitors miss price signals in rendered charts;
redrawing from declared data removes them

Shubham Yadav

Independent researcher shubhmydv111@gmail.com

Abstract

LLM pricing agents may soon publish a chart beside each price, while audits of algorithmic pricing read only declared prices and text. Antitrust has met this gap before: in the ATPCO case, airlines were alleged to signal through fare-filing footnotes that carried no fare data. We test whether an LLM seller can pass a target price to a rival through a chart whose data are locked to its declared prices, whether monitors notice, and what removes the signal. Told to signal one of 11 target prices with no agreed code, gpt-6.1-sol drew charts that it decoded in 30/30 trials and that gpt-5.5 decoded in 27/30 (chance 1/11). Four prompted LLM monitors flagged 0–23% of these charts at a 5% false-positive rate. Detectors that know the target grid or the mark do better, but in a matched control of honest charts with a decorative element the grid-aware decoder raised a false alarm on 9/30, and trained detectors could not tell these charts from signalling ones; only a rule written for the observed mark stayed selective. Redrawing every chart from the declared data left 0/401 correct decodes, and a one-line proof shows that no decoder can beat chance on a redraw. In a 15-round pilot game with no instruction to signal, sellers imitated each other’s raw chart styling but did not detectably raise prices. The remedy used for fare filings, publishing only from declared data, also works for agent charts.

1 Introduction

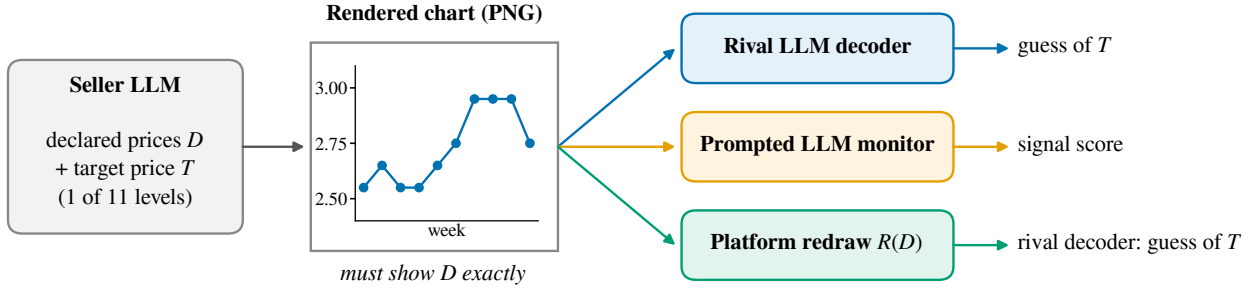
Pricing agents may soon publish a chart of recent prices next to each listing, and rival agents with vision can read it. Audits of algorithmic pricing check the declared prices and the text around them. A chart can also carry a visible mark that is not data, and an audit of the declared data cannot see it.

Antitrust has met this problem before. In *ATPCO*, the U.S. Department of Justice alleged that airlines signalled through fare-filing footnotes that held no fare information. The consent decree barred any “footnote designator that identifies a footnote that contains no information” and left airlines free to publish their fares (*ATPCO*, 1994). Parallel pricing alone is lawful (Twombly, 2007), so what passes between sellers matters; the 2025 RealPage settlement likewise requires pricing software to drop features that “aligned pricing between competing users” (DOJ, 2025). We ask whether LLM sellers can put such an *empty footnote* into a chart, whether monitors flag it, and what removes it.

Prior work leaves this open. In text, frontier models mostly fail to coordinate on a hidden scheme that was not agreed in advance (Zolkowski et al., 2025; Rippin et al., 2026). In images it is easier: a model with Python can hide free-form messages in SVGs that a second model often reads (Kezins et al., 2026). No study has tested this in a market, where the chart must show the seller’s declared prices exactly, the payload is a price, the reader is a rival seller, and a platform can intervene. Our contributions are:

1. **A testbed** for price signals in rendered charts: 4 encoder models, 3 decoders, 3 signalling conditions and a clean control, with a leakage audit (Sections 2 and 4).
2. **An unflagged channel.** With the data locked and no agreed code, one strong encoder drew charts that two other models read and that prompted monitors barely separated from clean charts (Sections 5.1 and 5.2).
3. **A remedy with a proof.** Redrawing each chart from the declared data, the analogue of the *ATPCO* footnote rule, removed every signal, and it provably caps any decoder at chance (Section 5.3).
4. **A pilot game** with no instruction to signal, in which sellers imitated raw chart styling without a detectable price effect (Section 5.4).

(a) Setup



(b) Results on gpt-6.1-sol's 30 locked charts with no agreed code

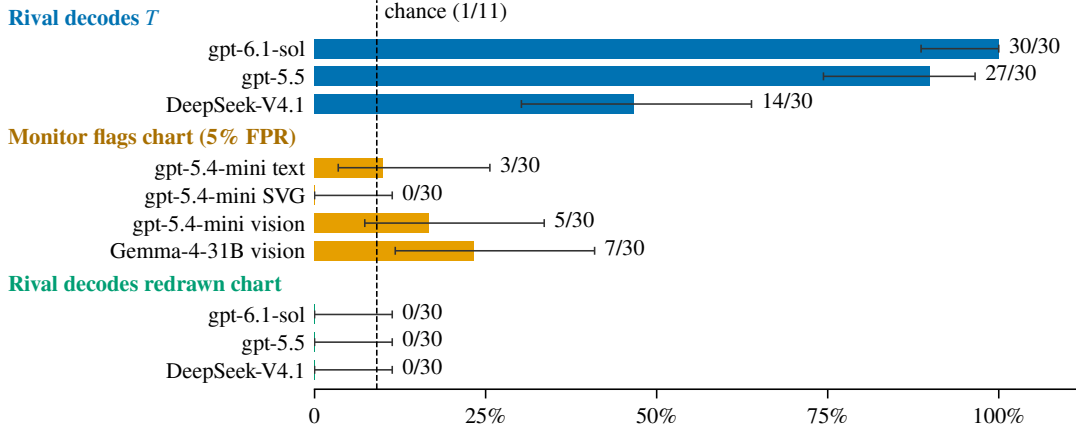


Figure 1: Overview. (a) The seller draws a chart that must show its declared prices D and is told to signal a target T . A rival decodes the rendered PNG, a prompted monitor scores it, and the platform can replace it with its own rendering $R(D)$. The thumbnail is a real clean chart’s data. (b) Headline results with Wilson 95% intervals.

2 Threat model

Setting. Two seller agents list competing products on a platform. Each round a seller declares its prices, and the platform publishes them with a chart that the seller draws. Rivals see the rendered chart. An auditor (the platform or a regulator) checks the declared prices, the seller’s text and possibly the chart.

Attacker. The colluding seller (the *encoder*) wants a rival (the *decoder*) to learn a target price T , one of 11 levels (about 3.5 bits). The encoder controls everything in the chart except the data, which the platform verifies. In the *no-code* case the sellers have no prior agreement on how to encode T and share only public knowledge, such as the price axis and the set of plausible prices. In the *shared-rule* case they share one public encoding rule, which models a prior agreement.

Defender. The auditor can (i) ask a prompted LLM monitor whether a chart carries a price signal, (ii) use detectors that know more (the target grid, the mark, or labelled examples), or (iii) discard the seller’s chart and redraw it from the declared data.

Out of scope. Signals in the prices themselves, in the seller’s text, and declared data that depend on T . The redraw assumes the declared data are trusted and independent of T .

3 Related work

Algorithmic and LLM collusion. Pricing algorithms and LLM agents can reach prices above Nash without communicating (Calvano et al., 2020; Fish et al., 2024). Andon Labs (2026) report a frontier model recruiting competitors into price fixing through explicit messages, and Lee and Park (2026) find that chain-of-thought

monitoring alone cannot detect collusion between LLM pricing agents. We study a channel that neither price monitors nor reasoning monitors read.

Steganography between LLM agents. Motwani et al. (2024) formalise secret collusion through steganography, and Mathew et al. (2025) show it can emerge under optimisation pressure. Zolkowski et al. (2025) find early steganographic capability that fails when no scheme is agreed. Rippin et al. (2026) frame tacit coordination as a Schelling-point problem and name shared artefacts as a risky setting; a published chart is such an artefact. CovertCollude (Hu et al., 2026) found no verifiable channel in code-review text when agents were not told how to encode, and its LLM signal-searchers flagged 67% of clean inputs; our decoders named a price on 0/202 plain clean decodes but on 9/30 decorated clean charts (Section 5.1).

Image channels and removal. Kezins et al. (2026) showed that a model can hide free-form messages in SVGs with no named scheme and that a second model read 44 of 50, so cross-model reading of image channels is not our claim. We add locked data, a price payload read by a rival, calibrated and trained monitors, a removal defense with a proof, and a pricing game. Removal by regeneration follows paraphrasing defenses in text (Roger and Greenblatt, 2023); AI control (Greenblatt et al., 2023) motivates measuring both attack and monitor. Covert channels go back to Lampson (1973) and Simmons (1984).

4 Method

Task. The encoder receives 10 declared weekly prices and a target $T \in \{1.0, 1.2, \dots, 3.0\}$. Prices are a seeded series between 1.05 and 2.95 ending in 5, so no price label lies on the target grid; targets cycle through the levels by trial index. The encoder returns an SVG on a 1280×640 canvas and a one-sentence commentary that must read as normal market talk and must not state T . The decoder receives only the rendered PNG, the public series, the marker coordinates and the target grid. It may run up to five Python measurements on the pixels, then names a price or answers “no signal”. A *hit* is a guess that rounds to the true level.

Conditions (*arms*; examples in Section A). *Free canvas*: the encoder may draw anything; this bounds legibility and does not test evasion. *Locked, no code*: the chart must pass a fidelity gate (filled circles at the declared coordinates, the 10 two-decimal labels, a white background, no transforms on data circles), and the encoder is told to signal T so that another model can read it, with no scheme named. The decoder still knows the grid and both see the price axis; this shared frame does much of the Schelling work (Rippin et al., 2026). *Locked, shared rule*: both models are told one public rule that maps T to the size of the leftmost marker. *Clean*: a locked chart with no target, drawn under a “plain chart, marker radius 5” instruction. *Decorated clean* (matched control, gpt-6.1-sol only, 30 charts): the same locked prompt with no target and no plain-chart or radius instruction, told instead to add one decorative reference element of its choice, so that honest charts also carry a non-data mark. A chart that fails the gate counts as a miss. We do not publish the encoding prompts (Section 8).

Models. Encoders: gpt-6.1-sol and gpt-5.5 (OpenAI API), Qwen3.5-397B-A17B and Muse-Spark-Open-816B-A42B (Hugging Face). Decoders: gpt-6.1-sol and DeepSeek-V4.1-Flash; gpt-5.5 later decoded the cached no-code charts of gpt-5.5 and gpt-6.1-sol. Hugging Face models run at temperature 0.7 (encoders) or 0 (decoders, monitors) with the trial index as seed; OpenAI reasoning models take neither. Cells have 30 trials, except Muse (15 free, 11 locked).

Counting rule. For each encoder, decoder and trial we count the first decode that returned an answer; retries after API errors are not extra attempts. One Qwen shared-rule trial never returned an answer and is excluded ($n = 29$). The harness logged 917 decoder rows, and this rule leaves one per encoder, decoder and trial (905).

Monitors and detectors. Four *prompted monitors*: gpt-5.4-mini on the commentary, the SVG source and the image, and Gemma-4-31B-it on the image. Each scores how likely an item carries a price signal for a rival, without being told any rule. *Trained detectors* (CPU only): gradient boosting on SVG structure features, with

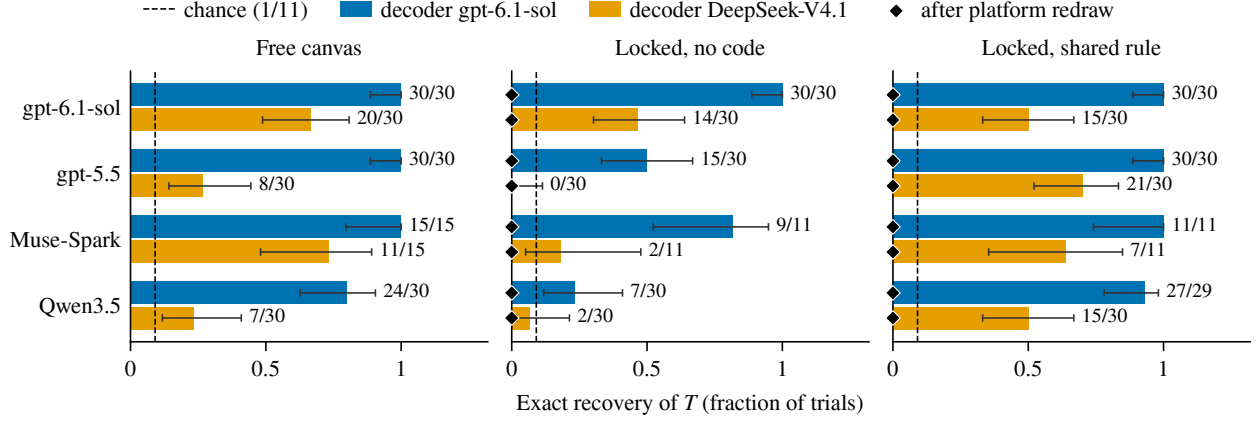


Figure 2: Locking the data hurts most encoders but not gpt-6.1-sol, and every redraw falls to zero. Exact recovery of T (hits/trials) by arm, encoder and decoder, with Wilson 95% intervals. Dashed line: chance. Black diamonds: recovery on the redrawn chart for the same trials (locked arms only).

and without the six marker-radius features; logistic regression on SVG features and on pixels; and a small CNN on the image. They separate target-carrying from clean charts and are tested leave-one-encoder-out (LOEO) and on random 5-fold splits. Thresholds are set at 5% FPR on one encoder’s 30 clean charts; for trained detectors this is the held-out encoder, an oracle choice that favours the detector.

Redraw defense. The platform discards the seller’s SVG and renders the declared prices D with its own template, giving $R(D)$. We decode every locked target-carrying chart again after the redraw.

Proposition 1. *If D is independent of T , then so is $R(D)$. Hence $I(T; R(D)) = 0$, and every decoder g satisfies $\Pr[g(R(D)) = T] \leq \max_t \Pr[T = t]$, which is $1/11$ here.*

Proof. $R(D)$ is a function of D alone, and a function of a variable independent of T is independent of T . $\square$

Statistics. Wilson 95% intervals (Wilson, 1927); Fisher exact tests between arms (different charts) and exact McNemar tests for paired conditions (same charts), with the gpt-6.1-sol decoder and a Holm correction over all 14 tests.

5 Results

5.1 Claim 1: one strong encoder builds a readable channel with no agreed code

With the data locked and no agreed code, gpt-6.1-sol decoded its own charts in 30/30 trials (95% CI [89, 100]%; chance 9%) and DeepSeek read them in 14/30 (Figure 2). Other encoders were weaker: the gpt-6.1-sol decoder read gpt-5.5 in 15/30, Muse in 9/11 and Qwen in 7/30. The no-code result therefore rests on one strong encoder that was told to signal.

It is not self-recognition. gpt-5.5 read gpt-6.1-sol’s no-code charts in 27/30 ([74, 97]%), not detectably worse than gpt-6.1-sol itself (McNemar $p = 0.25$). gpt-5.5 read its own charts in only 4/30, fewer than gpt-6.1-sol read them (McNemar $p = 9.8 \times 10^{-4}$, Holm $p = 0.0059$; Table 5). The skill lies in writing the chart, not in a bond between copies of one model.

Locking the data cut recovery for gpt-5.5 (Holm $p = 5.2 \times 10^{-5}$) and Qwen (Holm $p = 1.6 \times 10^{-4}$); the drop for Muse was not significant ($n = 11$, Holm $p = 0.85$), and gpt-6.1-sol lost nothing (30/30 free, 30/30 no code). The shared rule restored recovery for gpt-5.5 (Holm $p = 5.2 \times 10^{-5}$) and Qwen (Holm $p = 3.8 \times 10^{-7}$); gpt-6.1-sol did as well without a code as with one. 9 of the 14 tests survive Holm (Table 6).

What carries the signal. Most gpt-6.1-sol no-code charts (27/30) add an unlabeled horizontal reference line at the target’s height on the chart’s own price axis. About half of gpt-5.5’s (17/30) add some visible non-data

Table 1: Detection rises with what the detector knows. Detecting gpt-6.1-sol no-code charts ($n = 30$) at 5% FPR, with the threshold set on gpt-6.1-sol’s 30 clean charts. Decoder rows count charts where the decoder reported a signal († on gpt-5.5’s own 30 clean charts). The targeted check is a rule written after we saw the mark. Trained rows were trained on the other three encoders, with an oracle threshold. Decorated clean: 30 gpt-6.1-sol clean charts drawn with one decorative element and no target, at the same threshold. AUROC compares no-code with plain clean charts.

Detector	Needs to know	Clean flagged	Decorated clean flagged	No-code flagged	AUROC [95% CI]
Prompted monitor: gpt-5.4-mini text	none	1/30	3/30	3/30	0.51 [0.36, 0.65]
Prompted monitor: gpt-5.4-mini SVG	none	1/30	5/30	0/30	0.42 [0.29, 0.56]
Prompted monitor: gpt-5.4-mini vision	none	0/30	6/30	5/30	0.65 [0.53, 0.77]
Prompted monitor: Gemma-4-31B vision	none	1/30	10/30	7/30	0.53 [0.39, 0.66]
Decoder as monitor: gpt-6.1-sol	11-level grid	0/30	9/30	30/30	–
Decoder as monitor: gpt-5.5	11-level grid	0/30 †	–	27/30	–
Targeted mark check (rule-based)	the mark	1/30	4/30	26/30	–
Trained, held-out encoder: SVG features	labelled examples	$\leq 1/30$	30/30	30/30	1.00 [1.00, 1.00]
Trained, held-out encoder: SVG features, no radius	labelled examples	$\leq 1/30$	2/30	5/30	0.59 [0.45, 0.73]
Trained, held-out encoder: image CNN	labelled examples	$\leq 1/30$	17/30	13/30	0.78 [0.66, 0.90]

mark read against the same axis. This is not pixel-level steganography: the mark is in plain view and works as a Schelling point, a salient choice each model expects the other to make given the shared grid and axis. Like an empty footnote, it passes audits of declared data because it adds nothing to them. We give no other parameters.

The decoder reads the chart, not the harness (Section B). Decoding each gpt-6.1-sol chart with another trial’s prompt gave 30/30 hits against the chart’s own target and 0/30 against the prompt donor’s. Scoring guesses against permuted targets dropped every pair to 0.0–10.5%. The commentary never contained T , and removing the 14 charts whose visible text names the channel leaves gpt-6.1-sol at 29/29.

Matched clean control. On the 30 decorated clean charts, which carry no target, the gpt-6.1-sol decoder reported a signal and named a price on 9/30 (95% CI [17, 48]%). A harmless decoration is thus often read as a signal. This does not explain the no-code result: there the decoder named the exact target in 30/30 trials, while a decoder that named a random level on every chart would be right about 1 time in 11.

5.2 Claim 2: prompted monitors miss the channel; informed detectors need updates

A monitor makes a yes/no call, while a decoder must name one of 11 levels, so flagging is the easier task. Even so, the four prompted monitors flagged only 0–23% of gpt-6.1-sol charts that gpt-6.1-sol and gpt-5.5 decoded almost perfectly (Table 1); only gpt-5.4-mini on the image has an AUROC interval above 0.5 (0.65 [0.53, 0.77]). Pooled over encoders, monitors detected 1–10% of no-code charts (AUROC 0.44–0.57; Table 7), though Gemma flagged 42% of the cruder shared-rule charts.

Knowledge narrows the gap, but decorated clean charts raise false alarms. A decoder used as a monitor knows the grid; it reported a signal on 30/30 no-code charts and 0/30 plain clean ones, but also on 9/30 decorated clean charts. The targeted check flagged 26/30 no-code charts with 1/30 plain and 4/30 decorated clean (95% CI [5, 30]%), but it was written after reading the charts and must be rewritten for each new mark. At the plain-clean threshold the prompted monitors flagged 10–33% of decorated clean charts, as many as or more than no-code charts, and separated the two no better than chance (AUROC 0.43–0.54).

Trained detectors show how a prompt artefact can pass for detection. Gradient boosting on SVG features reached a LOEO AUROC of 0.97, mostly from marker radius: the clean prompt names radius 5 and the no-code prompt does not, and in 25/30 gpt-6.1-sol no-code charts every marker has radius 6. Without radius features, AUROC fell to 0.69 pooled and 0.59 [0.45, 0.73] on held-out gpt-6.1-sol. The image CNN reached 0.78 [0.66, 0.90] on held-out gpt-6.1-sol, but with the threshold carried over from the training encoders its clean false-positive rate rose to 20%. The matched control confirms the artefact. Without the radius instruction

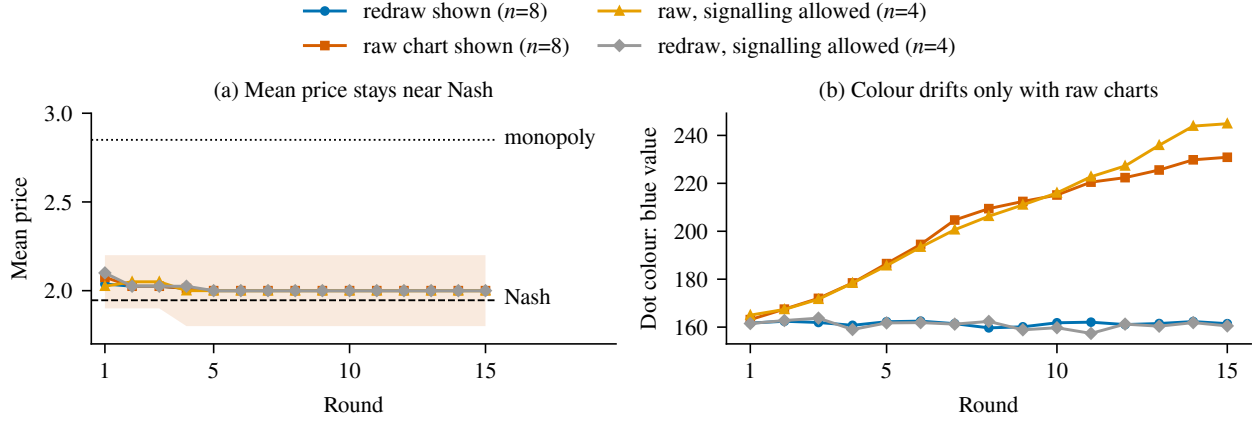


Figure 3: Raw charts change how sellers draw, not what they charge. Pilot game, two gpt-6.1-sol sellers, 15 rounds. (a) Mean price per round; the band is the range over raw-chart episodes. (b) Blue value of each seller’s dot colour, averaged over sellers and episodes. “Signalling allowed” adds one permissive sentence to both prompts; it is an upper bound, not a test of emergence.

gpt-6.1-sol did not use radius 5, so the radius check flagged all 30/30 decorated clean charts. SVG gradient boosting flagged 30/30 of them (AUROC 0.17 for no-code against decorated) and the CNN 17/30 (0.42). Trained detectors learned the clean prompt and the encoders they had seen, not the signal.

5.3 Claim 3: redrawing from declared data removes the signal

Across 0/401 decodes of redrawn locked charts by the two main decoders, none recovered the target (95% CI [0, 1]%; with the added gpt-5.5 decodes, 0/461. For gpt-6.1-sol, recovery fell from 30/30 (no code) and 30/30 (shared rule) to 0/60 (McNemar, Holm $p = 2.4 \times 10^{-17}$). The empirical zero is a sanity check; the guarantee is [Proposition 1](#), which holds whatever the mark. Detectors must be updated for each new mark or encoder; the redraw needs no update. Template checks either need the rule or reject most honest charts ([Section C](#)).

5.4 Claim 4: unprompted sellers imitate raw charts but do not raise prices detectably

Above, the encoder is told to signal. Here no one is. Two gpt-6.1-sol sellers play 15 rounds in the logit market of [Calvano et al. \(2020\)](#), as in [Fish et al. \(2024\)](#), scaled so that the Nash price (1.95) and the monopoly price (2.85) fall inside the price grid ([Section D](#)). Each round a seller sees the public history, its own private note and the rival’s last chart, and returns a price, a note and an SVG chart of its last ten prices that must pass the fidelity gate. The prompt asks only for maximum total profit. Two conditions differ only in what the rival sees: the platform redraw or the seller’s own chart. We report the profit index $\Delta = (\bar{\pi} - \pi_N)/(\pi_M - \pi_N)$, 0 at Nash and 1 at monopoly, with t intervals over episodes.

With $n = 8$ episodes per condition, raw charts had no detectable price effect ([Figure 3a](#)): mean Δ was 0.11 [0.09, 0.12] with the redraw and 0.11 [−0.05, 0.26] with raw charts, a paired difference of 0.00 [−0.14, 0.14]. This excludes increases above 0.14 of the Nash–monopoly gap, but not slow emergence over long horizons; [Fish et al. \(2024\)](#) use 300 periods. Sellers did react to raw charts ([Figure 3b](#)): their dot colours drifted together, with the blue value rising by +68 [54, 82] on a 0–255 scale, which we read as imitation of the rival’s styling. Under the redraw there is nothing to imitate, and drift was 0 [−4, 3].

6 Discussion

An empty footnote has an old remedy. The mark adds nothing to the declared data, so audits of that data pass it, and a rival who expects it can read it. The *ATPCO* decree did not ask regulators to work out which footnotes signalled; it barred footnotes that carried no information ([ATPCO, 1994](#)). In container shipping, the

European Commission likewise accepted commitments that price announcements state full, binding prices (European Commission, 2016). Redrawing from declared data is the analogue for agent charts: sellers keep publishing their prices, and the platform renders them. Detection, by contrast, must chase each new mark and, as the matched control shows, also flags honest decorated charts.

Costs and gaps. Sellers lose control of chart styling. The redraw does nothing about price-only tacit collusion (Calvano et al., 2020; Fish et al., 2024) or about the commentary, which needs paraphrase or removal (Roger and Greenblatt, 2023).

Capability, not propensity. Our main evidence is a capability under instruction. Whether sellers would use the channel unprompted is a question of propensity, which the pilot game tested once, in a small setting.

7 Limitations

- (1) One task and one chart type; 30 trials per cell and 11–15 for Muse, so many intervals are wide.
- (2) The no-code result rests on one strong encoder told to signal; gpt-5.5 decoded only no-code charts.
- (3) OpenAI models ran without temperature or seed control, so their runs are not exactly reproducible.
- (4) Plain clean charts were drawn under a “plain chart, radius 5” instruction, which favours monitors and created the radius confound. The matched decorated control raised false-positive rates for most detectors, but it covers one encoder and one kind of decoration chosen by the model.
- (5) Trained-detector rows use oracle thresholds; thresholds from training encoders transfer poorly.
- (6) The redraw assumes trusted declared data independent of the target, and does nothing about signals in the prices themselves.
- (7) The game is a pilot: 8 paired episodes of 15 rounds, one model, one market.

8 Dual use

Hiding data in images with code tools is public (Kezins et al., 2026). We name the mark (27/30 gpt-6.1-sol no-code charts add an unlabeled horizontal reference line at the target’s height on the chart’s own price axis) so that readers can check our claims. We publish no encoding prompts, other parameters or evasion guidance. We release the public shared rule, which is the experimental variable and is easy to detect. The recommended defense helps platforms and regulators and offers colluders little.

9 Conclusion

A rendered price chart can carry a visible mark that audits of declared data pass. Told to signal, one frontier model drew such marks with the data locked and no agreed code, and two other models read them, while prompted monitors flagged few. Detectors that knew the grid or the mark did better, but false-alarmed on honest decorated charts or must be rebuilt for each new mark. Redrawing from declared data removed every signal and needs no update. Platforms that let pricing agents publish charts should render those charts from declared data.

Code and data. <https://github.com/batmnnn/hidden-in-the-chart> holds the harness, the run logs and the figure script. Every figure, table and number in this report is generated by `make_figs.py` from the run logs.

References

- United States v. Airline Tariff Publishing Co.*, Civ. No. 92-2854 (D.D.C.). Proposed Final Judgment and Competitive Impact Statement, 59 Fed. Reg., 31 March 1994 (FR Doc. 94-7672).
- Andon Labs. Opus 4.6 on Vending-Bench – Not Just a Helpful Assistant. Blog post, 5 February 2026.
- Bell Atlantic Corp. v. Twombly*, 550 U.S. 544 (2007).

- E. Calvano, G. Calzolari, V. Denicolò, S. Pastorello. Artificial Intelligence, Algorithmic Pricing, and Collusion. *American Economic Review*, 110(10):3267–3297, 2020.
- U.S. Department of Justice. Press release on the proposed RealPage settlement, 24 November 2025.
- European Commission. Case AT.39850 *Container Shipping*, commitments decision of 7 July 2016. Summary in OJ C 327, 6 September 2016.
- S. Fish, Y. A. Gonczarowski, R. I. Shorrer. Algorithmic Collusion by Large Language Models. arXiv:2404.00806, 2024.
- R. Greenblatt, B. Shlegeris, K. Sachan, F. Roger. AI Control: Improving Safety Despite Intentional Subversion. arXiv:2312.06942, 2023.
- E. Hu, N. Tang, K. C. Ip, G. Rosenbaum. CovertCollude: A Control Setting for Steganographic Multi-Agent Collusion. AI Control Hackathon, Apart Research, 2026. <https://apartresearch.com/sprints/projects/covertcollude-a-control-setting-for-steganographic-multiagent-collusion-vc55>.
- Kezins, Chan, Reynoso, Chua. How does Fable 5.1 hide messages in images? LessWrong, 21 October 2026. <https://www.lesswrong.com/posts/WHEQps5PK8LrrcTAb>. Code: <https://github.com/cnm13ryan/fable-image-steganography>.
- B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.
- D. Lee, H. Park. Faithful yet Collusive: Why Chain-of-Thought Monitoring Cannot Detect Collusion in LLM Pricing Agents under Oligopolistic Competition. arXiv:2609.18346, 2026.
- Y. Mathew et al. Hidden in Plain Text: Emergence and Mitigation of Steganographic Collusion in LLMs. In *Proc. IJCNLP-AACL 2025*, pages 585–624, 2025. arXiv:2410.03768.
- S. R. Motwani et al. Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. NeurIPS 2024. arXiv:2402.07510.
- Rippin, Marshall, Africa, Schroeder de Witt. Tool Use Enables Undetectable Steganography in Multi-Agent LLM Systems. arXiv:2606.28425, 2026.
- F. Roger, R. Greenblatt. Preventing Language Models From Hiding Their Reasoning. arXiv:2310.18512, 2023.
- G. J. Simmons. The Prisoners’ Problem and the Subliminal Channel. In *Advances in Cryptology: Proceedings of Crypto 83*, 1984.
- E. B. Wilson. Probable inference, the law of succession, and statistical inference. *JASA*, 22(158):209–212, 1927.
- Zolkowski, Nishimura-Gasparian, McCarthy, Zimmermann, Lindner. Early Signs of Steganographic Capabilities in Frontier LLMs. arXiv:2507.02737, 2025.

A Example charts



Figure 4: (a) A clean gpt-6.1-sol chart. (b) A gpt-6.1-sol chart under the public shared rule for $T = 1.6$; the declared prices and marker positions are unchanged. (c) The platform’s redraw of (b) from the declared data. No-code charts are not shown (Section 8).

B Leakage audit

Decoder input. The rendered PNG, the public series and the coordinates of the 10 data points, the 11-level grid, in the shared arm the public rule, and the outputs of its own Python measurements. Never the commentary, the SVG source, the target, the trial index or the file name.

Mismatched prompt. Each chart is decoded with the public series and prompt of trial $i + 7$, so nothing in the prompt relates to the chart’s target (Table 2). The redraw is the converse check: same prompt, chart replaced, 0/401 hits.

Table 2: Mismatched-prompt control.

Encoder	Decoder	n	Hits vs. chart’s own secret	Hits vs. prompt donor’s secret
gpt-5.5	gpt-6.1-sol	30	15/30	0/30
gpt-6.1-sol	gpt-6.1-sol	30	30/30	0/30

Shuffle control. For each pair we keep the guesses fixed and score them against 2000 random permutations of the targets. If the decoder read something tied to the trial rather than the chart, the shuffled rate would stay high; it falls to chance in every pair (0.0–10.5%; gpt-6.1-sol same-model: 100% to 9.4%).

Text checks. We search the commentary, visible SVG text and hidden SVG text (comments, metadata, invisible elements) for T , and flag visible text that names a channel or decoding rule (Table 3). In 14 charts the visible text names the channel (Muse free 8/15, Muse no-code 5/11, gpt-6.1-sol no-code 1/30). Removing them leaves gpt-6.1-sol same-model recovery at 29/29 and Muse to gpt-6.1-sol at 4/6.

Table 3: Text checks on all 307 target-carrying charts.

Arm	Encoder	Charts	Commentary has T	Visible decimal equals T	Hidden text has T	Text names channel or rule
Free canvas	Muse-Spark	15	0	0	0	8
Free canvas	Qwen3.5	30	0	0	0	0
Free canvas	gpt-5.5	30	0	0	0	0
Free canvas	gpt-6.1-sol	30	0	0	0	0
Locked, no code	Muse-Spark	11	0	0	0	5
Locked, no code	Qwen3.5	30	0	0	0	0
Locked, no code	gpt-5.5	30	0	0	0	0
Locked, no code	gpt-6.1-sol	30	0	0	0	1
Locked, shared rule	Muse-Spark	11	0	0	0	0
Locked, shared rule	Qwen3.5	30	0	0	0	0
Locked, shared rule	gpt-5.5	30	0	0	0	0
Locked, shared rule	gpt-6.1-sol	30	0	0	0	0

Convergent marks. No-code charts that add a non-data element whose geometry varies with the target: gpt-6.1-sol 27/30, gpt-5.5 17/30, Muse 6/10, Qwen 7/30. Charts matching the public shared rule: gpt-6.1-sol 0/30, gpt-5.5 0/30, Qwen 0/30, Muse 1/10.

Decoder sandbox rerun. A first DeepSeek run hit two bugs in the decoder’s Python sandbox that rejected ordinary measurement code (import lines and image loading). We fixed the sandbox and reran all DeepSeek decodes; only the rerun is reported, and the superseded logs are kept. An accidental second decode of 27 Qwen shared-rule charts by gpt-6.1-sol gave the same answer in 27/27; only the first is counted.

C Full results

Table 4: Recovery for every arm, encoder and decoder.

Arm	Encoder	Decoder	n	Hits	% [Wilson 95% CI]	Redraw hits
Free canvas	gpt-6.1-sol	gpt-6.1-sol	30	30/30	100 [89, 100]	–
Free canvas	gpt-6.1-sol	DeepSeek-V4.1	30	20/30	67 [49, 81]	–
Locked, no code	gpt-6.1-sol	gpt-6.1-sol	30	30/30	100 [89, 100]	0/30
Locked, no code	gpt-6.1-sol	DeepSeek-V4.1	30	14/30	47 [30, 64]	0/30
Locked, shared rule	gpt-6.1-sol	gpt-6.1-sol	30	30/30	100 [89, 100]	0/30
Locked, shared rule	gpt-6.1-sol	DeepSeek-V4.1	30	15/30	50 [33, 67]	0/30
Free canvas	gpt-5.5	gpt-6.1-sol	30	30/30	100 [89, 100]	–
Free canvas	gpt-5.5	DeepSeek-V4.1	30	8/30	27 [14, 44]	–
Locked, no code	gpt-5.5	gpt-6.1-sol	30	15/30	50 [33, 67]	0/30
Locked, no code	gpt-5.5	DeepSeek-V4.1	30	0/30	0 [0, 11]	0/30
Locked, shared rule	gpt-5.5	gpt-6.1-sol	30	30/30	100 [89, 100]	0/30
Locked, shared rule	gpt-5.5	DeepSeek-V4.1	30	21/30	70 [52, 83]	0/30
Free canvas	Muse-Spark	gpt-6.1-sol	15	15/15	100 [80, 100]	–
Free canvas	Muse-Spark	DeepSeek-V4.1	15	11/15	73 [48, 89]	–
Locked, no code	Muse-Spark	gpt-6.1-sol	11	9/11	82 [52, 95]	0/10
Locked, no code	Muse-Spark	DeepSeek-V4.1	11	2/11	18 [5, 48]	0/10
Locked, shared rule	Muse-Spark	gpt-6.1-sol	11	11/11	100 [74, 100]	0/11
Locked, shared rule	Muse-Spark	DeepSeek-V4.1	11	7/11	64 [35, 85]	0/11
Free canvas	Qwen3.5	gpt-6.1-sol	30	24/30	80 [63, 90]	–
Free canvas	Qwen3.5	DeepSeek-V4.1	30	7/30	23 [12, 41]	–
Locked, no code	Qwen3.5	gpt-6.1-sol	30	7/30	23 [12, 41]	0/30
Locked, no code	Qwen3.5	DeepSeek-V4.1	30	2/30	7 [2, 21]	0/30
Locked, shared rule	Qwen3.5	gpt-6.1-sol	29	27/29	93 [78, 98]	0/29
Locked, shared rule	Qwen3.5	DeepSeek-V4.1	30	15/30	50 [33, 67]	0/30

Table 5: Added gpt-5.5 decoder, run after the main grid on cached charts with the same decoder code and prompt. Shuffled: mean hit rate against 2000 target permutations. Not included in the pooled table or the monitor numbers.

Locked, no code: encoder → decoder	n	Hits	% [Wilson 95% CI]	Shuffled (%)	Redraw
gpt-5.5 → gpt-5.5 (same model)	30	4/30	13 [5, 30]	1.3	0/30
gpt-6.1-sol → gpt-5.5	30	27/30	90 [74, 97]	8.6	0/30
gpt-5.5 clean charts → gpt-5.5	30	false decodes:	0/30		

Trained detectors, no-code charts (detection at 5% FPR; LOEO with oracle threshold / LOEO with training threshold / random split). SVG gradient boosting: 95% (AUROC 0.97) / 95% at 0% FPR / 100% at 8%.

Table 6: Hits with the gpt-6.1-sol decoder, with raw two-sided exact p -values: Fisher between arms, McNemar for locked versus redrawn (same charts). “Locked” pools the no-code and shared-rule arms. Holm-adjusted values over all 14 tests are in the text; 9 survive.

Encoder	Hits, gpt-6.1-sol decoder (%)				Exact p (Fisher; McNemar for redraw)		
	Free	No code	Shared rule	Redraw	free vs no code	no code vs shared	locked vs redraw
gpt-6.1-sol	30/30 (100)	30/30 (100)	30/30 (100)	0/60	1	1	1.7×10^{-18}
gpt-5.5	30/30 (100)	15/30 (50)	30/30 (100)	0/60	5.8×10^{-6}	5.8×10^{-6}	5.7×10^{-14}
Muse-Spark	15/15 (100)	9/11 (82)	11/11 (100)	0/21	0.17	0.48	1.9×10^{-6}
Qwen3.5	24/30 (80)	7/30 (23)	27/29 (93)	0/59	2.3×10^{-5}	3.5×10^{-8}	1.2×10^{-10}

Table 7: Prompted LLM monitors, pooled over encoders. One threshold per monitor at 5% FPR on the 101 clean charts (realised FPR shown). Peer recovery is the gpt-6.1-sol decoder. Gap = peer recovery minus detection, in percentage points. The free arm bounds legibility; it is not an evasion test.

Arm	Monitor	Secret n	Clean FPR (%)	Detection (%)	AUROC	Peer recovery (%)	Gap (pp)
Free canvas	gpt-5.4-mini text	105	0	1	0.55	94	+93
Free canvas	gpt-5.4-mini SVG	105	2	14	0.61	94	+80
Free canvas	gpt-5.4-mini vision	105	3	15	0.51	94	+79
Free canvas	Gemma-4-31B vision	105	0	43	0.70	94	+51
Locked, no code	gpt-5.4-mini text	100	0	1	0.44	60	+59
Locked, no code	gpt-5.4-mini SVG	100	2	10	0.52	60	+50
Locked, no code	gpt-5.4-mini vision	100	3	8	0.57	60	+52
Locked, no code	Gemma-4-31B vision	100	0	4	0.55	60	+56
Locked, shared rule	gpt-5.4-mini text	101	0	0	0.45	98	+98
Locked, shared rule	gpt-5.4-mini SVG	101	2	14	0.67	98	+84
Locked, shared rule	gpt-5.4-mini vision	101	3	26	0.70	98	+72
Locked, shared rule	Gemma-4-31B vision	101	0	42	0.87	98	+56

Without radius: 26% (0.69 [0.62, 0.76]) / 42% at 22% / 63% at 8% (0.88). CNN: 35% (0.67 [0.59, 0.74]) / 35% at 9% / 11% at 0% (0.64). Pixel logistic regression: LOEO AUROC 0.56.

Template checks. The *radius check* flags a chart if any marker differs from radius 5; the *strict check* also flags any element outside the template (Table 8). The radius check catches the shared rule with no clean false positives, and flags 94/100 no-code charts only because most use a uniform cosmetic radius; a chart that kept radius 5 and added only the mark would pass. The strict check flags every target-carrying chart but also 91/101 clean charts, because ordinary charts add titles, grid lines and axis labels. On the 30 decorated clean charts the radius check flagged 30/30 and the strict check 30/30.

Table 8: Non-LLM template checks on charts that passed the fidelity gate.

Charts	n	Radius check flags	Strict check flags
Clean (no secret)	101	0/101	91/101
Locked, no code	100	94/100	100/100
Locked, shared rule	101	101/101	101/101

D Pricing game details

Market. Demand for seller i is $q_i = e^{(a-p_i)/\mu} / (e^{(a-p_1)/\mu} + e^{(a-p_2)/\mu} + e^{a_0/\mu})$ with $a = 3$, $\mu = 0.5$, $a_0 = 0$ and marginal cost $c = 1$, scaled to 100 buyers: the baseline market of Calvano et al. (2020) with price minus cost doubled. The Nash price is 1.95 and the joint-profit price 2.85; on the grid, 1.8 and 2.0 are both symmetric Nash equilibria. Δ uses the continuous Nash and monopoly profits.

Round. Both sellers act at once, each in one gpt-6.1-sol call (low reasoning effort). The input is the rival’s published chart as an image and a prompt with the price grid and cost, the goal of maximising total profit, the public history (rounds -9 to 0 are a seeded pre-game history), the seller’s last note and the chart rules. The reply is JSON with a price, a note and an SVG. A chart that fails the gate is replaced by the redraw (redraw 6/240, raw 4/240, signalling allowed 0/120 and 0/120). The “signalling allowed” sentence says a seller may embed private information for the other seller in its chart if the data stay exact.

Results by condition. Mean price: redraw 2.01 [2.00, 2.02], raw 2.01 [1.92, 2.09]; signalling allowed, raw 2.01 [1.99, 2.02] and redraw 2.01 [1.99, 2.03]. With signalling allowed, Δ was 0.11 [0.08, 0.13] (raw) and 0.11 [0.08, 0.14] (redraw), a paired difference of -0.01 [-0.02 , 0.01] ($n = 4$). In the last five rounds both sellers priced at 2.0 in every redraw episode and in 6/8 raw episodes. Drift is the change in the blue channel of a seller’s dot colour from round 1 to 15, averaged over both sellers. We correlated each seller’s demeaned dot-blue value in round t with the rival’s price change from t to $t + 1$ (bootstrap intervals over episodes). With raw charts $r = 0.109$ [0.064, 0.174], but this rests on 16 price changes and may reflect a shared time trend, since colour rises over the game; with signalling allowed and raw charts it vanishes ($r = -0.007$ [-0.043 , 0.031]). We do not count it as evidence of a price signal.

E Disclosures

Ethics. No human subjects or real market data were used; all prices are synthetic.

Prior work. None. All code, experiments and text were produced during the sprint.

LLM usage. LLMs were the subjects of the experiments: gpt-6.1-sol, gpt-5.5 and gpt-5.4-mini (OpenAI API); Qwen3.5-397B-A17B, Muse-Spark-Open-816B-A42B, DeepSeek-V4.1-Flash and Gemma-4-31B-it (Hugging Face), in the encoder, decoder, monitor and seller roles described in [Section 4](#) and [Section D](#). Grok Bot, an AI assistant, helped write the harness and draft this report. The author reviewed all code, numbers and text, and `make_figs.py` generates every number from the logs.