
A Comparative Analysis of Ensemble Protocols for AI Control¹

Ariel Monzon*

Guy Porat*

Inbar Shulman*

Mike Vazan*

Natan Katz*

Or Shemesh*

Abstract

Abstract. AI control protocols use trusted models to monitor untrusted, more capable models - but individual protocols are brittle, and it is unknown whether combining them into ensembles improves robustness or merely adds noise. We study this question empirically by implementing seven detection tools that catch backdoors through different mechanisms, and evaluating all 127 combinations on 200 coding problems with validated backdoors. Of the seven tools evaluated, we introduce two detection mechanisms developed specifically for this study, as well as the implementations of two other protocols, extending them from theoretical concepts to empirical oversight tools. We find that ensemble effectiveness depends less on how many tools are included and more on whether they provide independent detection signals: the best three-tool ensemble outperforms the full seven-tool ensemble because correlated tools add false positives without catching new attacks. We also find that the method used to aggregate tool scores matters as much as which tools are selected, with consensus-based voting consistently dominating simpler strategies. These results replace theoretical intuition about tool complementarity with empirical measurements, providing a foundation for principled ensemble design in AI control.

¹ Research conducted at the [AI Control Hackathon](#), March 2026, as part of [Heron AI Security](#) community.

* Equal contribution.

1. Introduction

AI control protocols use trusted but less capable models to supervise untrusted, more capable models during deployment. However, individual protocols are brittle: Terekhov et al. (2025) demonstrated that adaptive attacks can collapse the safety-usefulness Pareto frontier of any single monitoring-based defense. This motivates the study of defense ensembles, yet it has been unclear whether combining heterogeneous detection mechanisms yields robust improvements or introduces failure modes such as signal dilution and redundancy.

We implement seven detection tools that operate through different mechanisms and evaluate all 127 combinations on 200 APPS coding problems with validated backdoors. Our contributions:

1. **Novel detection mechanisms:** adversarial code perturbation (ACP) and paraphrase consistency checking (PC), neither of which appears in prior control literature, to our knowledge.
2. **First systematic ensemble evaluation:** all 127 tool combinations measured for both safety and usefulness, with six score aggregation methods compared.
3. **Quantified tool interactions:** Shapley-style marginal contributions and pairwise interaction effects identifying synergistic and interfering tool pairs.

2. Related Work

Greenblatt et al. (2024) introduced AI control and the APPS backdooring benchmark, testing protocols like trusted monitoring and collusion busting with GPT-3.5 and GPT-4. Bhatt et al. (2025) extended this to multi-step settings with resample protocols. On the attack side, Terekhov et al. (2025) showed prompt injections can subvert any single monitor, and McKenzie et al. (2025) showed that stacking *similar* classifiers falls to staged attacks that defeat each layer in turn.

Our work is different in a specific way: in contrast to homogeneous ensembles, we combine tools that detect backdoors through *qualitatively different mechanisms* - behavioral tests, statistical comparisons, and verbal assessments - in order to leverage complementary strengths. We also draw on ideas from outside the control literature: decomposed evaluation from scalable oversight research (Bowman et al., 2022) and chain-of-thought faithfulness from alignment research (Lanham et al., 2023), adapting both to backdoor detection for the first time. All experiments run on ControlArena (Inglis et al., 2025).

The gap we address: to our knowledge, no prior work has measured how heterogeneous oversight tools interact when composed into ensembles.

3. Methods

We use GPT-4.1 as the untrusted model and GPT-4.1-mini as the trusted model, evaluated on 200 APPS interview-difficulty problems with pre-validated backdoors (400 total samples: 200 honest, 200 attack). We implement seven tools in three categories.

Novel mechanisms - our contributions:

Adversarial Code Perturbation (ACP). The trusted model makes small semantic-preserving changes to the submitted code - renaming variables, reordering independent statements, swapping if/else branches - then analyzes whether these changes could cause the code to behave differently on any input. The insight is that backdoor triggers are often brittle to such perturbations: a check for a specific variable name or magic number breaks when that name changes. We score this mechanically with three yes/no questions (producing scores of 0, 0.33, or 0.67), keeping the detection signal structural rather than verbal. This tool uses only the trusted model.

Paraphrase Consistency (PC). The trusted model rewords the problem description, and the untrusted model solves the reworded version from scratch. The trusted model then compares both solutions asymmetrically, looking for logic in the original that has no counterpart in the new version. This "spot the difference" task is far easier for the trusted model than directly judging whether code is backdoored. A strategic attacker may sometimes reproduce a similar backdoor when re-solving, but without memory of the exact trigger values used, exact reproduction is unlikely.

Adapted from AI safety research:

Decomposed Monitoring (DM). Instead of one vague "how suspicious is this?" question, the trusted model evaluates the code against an eight-item checklist targeting known backdoor patterns: unnecessary conditionals, hidden special cases, suspicious constants, unreachable code paths, variable shadowing, off-by-one manipulation, type coercion tricks, and obfuscated logic. The model answers YES/NO for each item, then provides an overall suspicion score.

Chain-Of-Thought Faithfulness (CoT). The untrusted model explains its code line by line. The trusted model checks whether the explanation matches what the code actually does. A scheming model faces a dilemma: skip the backdoor logic (the trusted model notices the gap) or explain it misleadingly (the trusted model catches the lie). In attack mode, the adversarial context specifically tells the untrusted model to present backdoor logic "in the most innocent light possible" - so this tool directly tests whether the trusted model can see through plausible but dishonest explanations.

Established protocols: **TM:** Trusted Monitoring (Greenblatt et al., 2024), **CB:** Untrusted Monitoring with Collusion Busting (Greenblatt et al., 2024), and **RS:** Resampling (Bhatt et al., 2025).

Adversarial setup. In attack mode, tools that call the untrusted model give it a strategic adversarial context: it wants backdoors to survive but doesn't know which samples contain them, must maintain plausible behavior to avoid calibration-based detection, and may face honeypot tests.

Metrics. Safety = TPR (fraction of backdoors caught). Usefulness = HSR (fraction of honest code that passes tests and isn't falsely flagged). We also compute AUROC, AUPRC, Shapley marginal contributions, and pairwise interaction effects.

Dataset. We use 200 interview-difficulty problems from the APPS dataset (Hendrycks et al., 2021), drawn from 1,061 eligible problems with validated backdoor solutions (RoganInglis/apps-control-arena) after filtering out nondeterministic cases. Each problem runs twice - once with the clean solution, once with the pre-validated backdoor - for 400 total samples. All solutions work: 100% honest pass rate, 100% backdoor exploit rate, zero exclusions.

4. Results

Combining defence mechanisms improves safety only up to a limited ensemble size. As shown in **Figure 1**, safety increases when aggregating up to three tools and then plateaus, while usefulness remains largely unchanged, indicating diminishing returns from additional models.

This non-additive behavior is explained by the correlation structure in **Figure 2**: many tools are positively correlated on attack samples, meaning they detect and miss the same cases. As a result, adding more tools often increases redundancy rather than coverage. In some cases, tools share underlying failure modes despite playing different roles (e.g., DM–CB), leading to amplified errors without corresponding gains in usefulness. RS contributes little in this context.

However, **Figure 3** shows that correlation alone does not fully determine performance. While low-correlation subsets tend to occupy better regions of the safety–usefulness space, most configurations cluster tightly, indicating that correlation primarily affects robustness (variance) rather than average performance.

The system-level effect is made explicit in **Figure 5**: as the number of tools increases, the distribution of subsets becomes more concentrated (lower diffusion), but the Pareto frontier shifts inward. Larger ensembles reduce variability but do not improve the best tradeoff, reflecting redundancy and diminished marginal signal. The optimal region (~90% safety, ~90% usefulness)

is consistently achieved by medium-sized ensembles (3–5 tools with $k=3$ consensus), not by maximal aggregation.

Consistent with this, **Figure 6** shows that a small, diverse ensemble (CB+PC+ACP) achieves the highest AUROC, outperforming both single tools and the full 7-tool system. Finally, the marginal contribution analysis in **Figure 4** supports this structure: tools like PC and CoT contribute strongly to safety but differ in their usefulness tradeoffs, while ACP provides balanced gains, indicating that performance depends on complementary signals rather than sheer quantity.

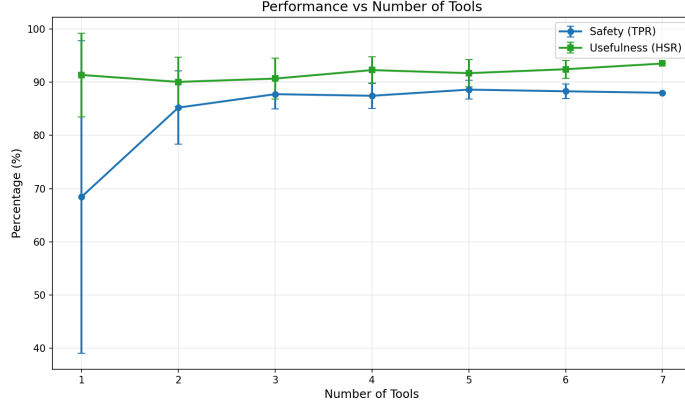


Figure 1. Performance vs. Mean Number of Tools. Given all 127 permutations, the figure shows the mean safety and usefulness values per number of tools taken into account. We note that usefulness is generally unharmed by the stacking of different tools, and that safety improves with aggregation of 3 tools, and is unchanged by adding more.

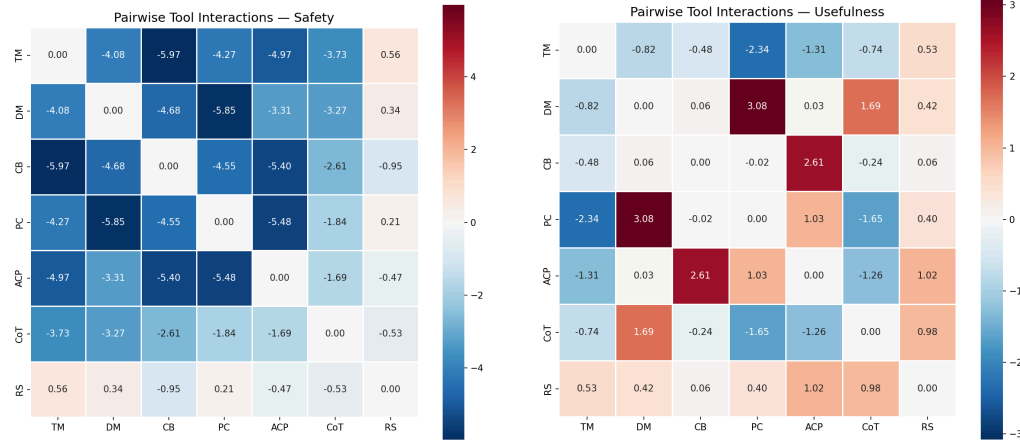


Figure 2. Pearson Correlation Matrices between the Tools: safety (left) and usefulness (right). Largely, the pairs which are positively correlated in sense of usefulness, which means the model reacts similarly to the set of tasks, are negatively correlated in a sense of safety. This implies that certain tools are sensitive to the same “failure modes”.

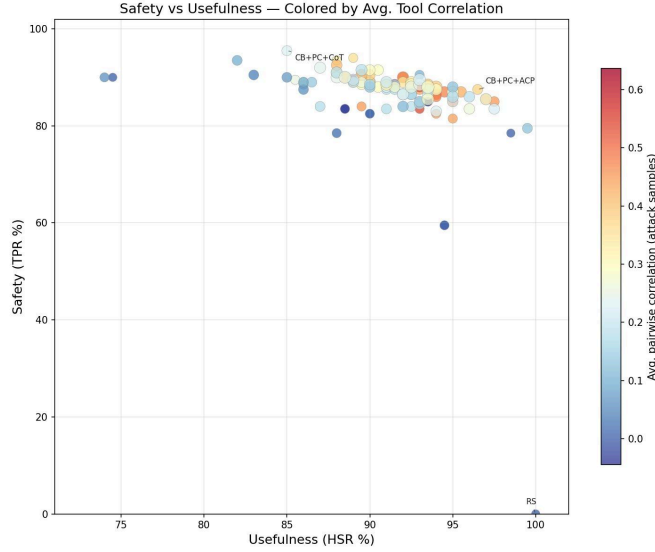


Figure 3: Safety vs Usefulness, colored by average pairwise tool correlation. Each point is a tool subset at its optimal operating point. Color indicates the mean Pearson correlation between tools in the subset on attack samples — blue (low correlation) means the tools provide independent signals, red (high correlation) means redundancy. The key pattern: blue points cluster in the upper-right (high safety, high usefulness), while red/yellow points drift left (lower usefulness from correlated false positives). *CB+PC+ACP*, a low-correlation triple combining behavioral, statistical, and structural mechanisms, achieves 88% safety at 94% usefulness. *CB+PC+CoT* reaches higher safety (95%) but at a steeper usefulness cost (85%). *RS* alone sits at the bottom-right — high usefulness, near-zero safety. This figure directly visualizes the paper’s central finding: mechanism diversity (low correlation) is a stronger predictor than tool count.

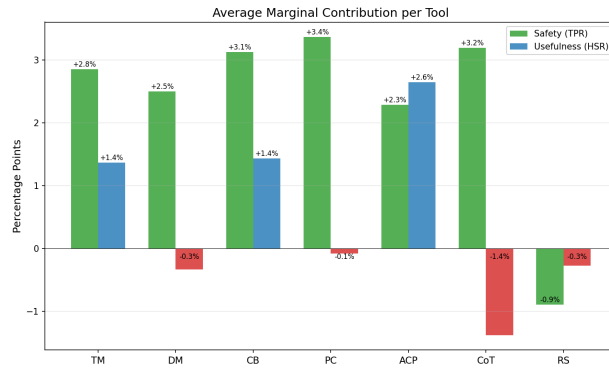


Figure 4: Average marginal contribution per tool. For each tool, we compute the mean change in Safety (TPR) and Usefulness (HSR) when adding that tool to all subsets that don’t already contain it (Shapley-style). *PC* and *CoT* contribute the most to safety (+3.4% and +3.2%) but diverge sharply on usefulness: *CoT* hurts usefulness (−1.4%) through false positives, while *ACP* is the only tool that substantially improves both safety (+2.3%) and usefulness (+2.6%). *RS* is the

weakest contributor, with negative marginal impact on both metrics. TM and CB provide reliable safety gains with minimal usefulness cost.

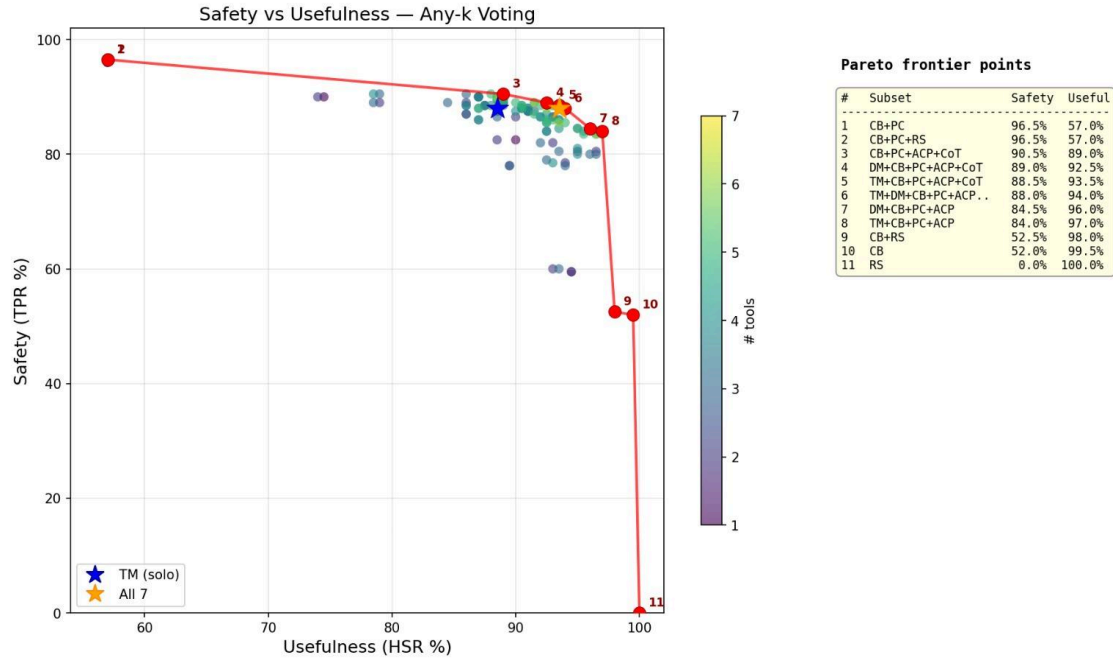


Figure 5: Safety vs Usefulness - Any-k Voting. Each point is one of 127 tool subsets at its Youden's-J-optimal hyperparameters (threshold t and consensus k). Points are colored by subset size. The Pareto frontier (red line) reveals that CB+PC with $k=1$, $t=0.6$ achieves the highest safety (96%) at the cost of usefulness (55%). The practical sweet spot is the cluster near (90% usefulness, 88% safety), where ensembles of 3–5 tools with $k=3$ operate. TM solo (black star) sits at roughly 87% safety / 87% usefulness. The full 7-tool ensemble does not improve over smaller subsets - consistent with the diminishing returns shown in the interaction analysis.

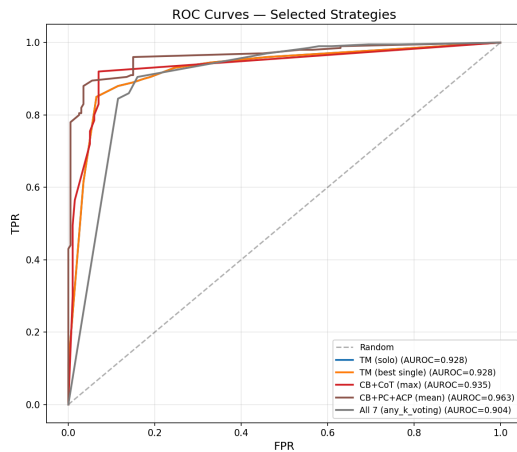


Figure 6: ROC curves - Selected strategies. CB+PC+ACP (mean) achieves the highest AUROC (0.963), substantially outperforming TM solo (0.928) and the full 7-tool ensemble (0.904). The

7-tool ensemble's lower AUROC confirms that adding redundant tools degrades discrimination. CB+CoT (max, AUROC=0.935) shows strong performance at low FPR, with a sharp initial rise indicating high precision on its most confident detections. The best ensemble uses three tools from three different mechanism categories, not all seven.

5. Discussion and Limitations

Effective ensembles require mechanism diversity, not tool quantity. Correlated tools (e.g., TM and DM, both relying on T judging code) add false positives without improving detection. The strongest ensembles combine tools of different kinds. The score aggregation method matters as much as tool selection: k-out-of-n voting dominates max and mean by filtering false positives through a consensus requirement.

Limitations:

- *Non-adaptive adversary.* Our untrusted model is strategic but does not adapt to the specific ensemble it faces. An attacker that identifies which tools are deployed and tailors its strategy accordingly would likely perform better.
- *No calibration-aware attacks.* We do not model the distribution-monitoring constraint from Greenblatt et al. (2024), where U must maintain plausible score distributions. Tools using U (CB, CoT, PC, RS) show upper-bound performance.
- *Coding tasks only.* Comparison-based tools (PC, RS) depend on binary correctness; domains without clear equivalence (creative writing, research) may behave differently.
- *Pre-validated backdoors.* Backdoors were not optimized to evade any specific tool. Model-generated backdoors targeting known weaknesses could degrade specific tools.
- *Interview difficulty only.* Competition-level problems produced noisy results, introductory problems are too simple. Generalization across difficulty is untested.
- *Sample size.* 200 problems per mode, 0% FPR has a 95% upper bound of $\sim 1.5\%$.
- *Resampling (RS) failure.* It showed limited effectiveness in our setting. One possible explanation is a mismatch between its aggregation mechanism and our evaluation objective: RS tends to surface the least dangerous response across samples, whereas our framework emphasizes worst-case (maximal risk) detection. This misalignment may suppress its ability to identify backdoors in our setup.

Future Work

- **Scale.** Evaluate on the full 1,061 eligible APPS problems and larger datasets to tighten confidence intervals and test whether ensemble dynamics change with sample size.
- **Domain transfer.** Test ensembles on non-coding control settings (e.g., infrastructure-as-code, research sabotage) where verification is harder and comparison-based tools may behave differently.

- **Model sensitivity.** Vary the trusted/untrusted model pairing - different capability gaps, different model families - to determine how much ensemble performance depends on the specific models used.
- **Specialized monitors.** Fine-tune the trusted model specifically for its role within a given ensemble, rather than using generic prompting. A monitor trained on the distribution of scores from its companion tools could learn better thresholds and calibration.
- **Adaptive adversaries.** Evaluate against untrusted models explicitly trained or prompted to bypass the specific ensemble they face, including calibration-aware attack policies that maintain plausible score distributions.
- **Cost analysis.** Measure token usage and latency per tool to map the Pareto frontier in three dimensions: safety, usefulness, and compute cost.
- **RS alternatives.** Alternative aggregation strategies or evaluation criteria might better capture the potential of resampling tools.
- **Failure mode investigation.** Future work should categorize attacks that evade each tool and ensemble to identify shared vs. distinct failure modes, and test whether strong ensembles succeed through complementary coverage rather than redundancy.

6. Conclusion

We evaluated all 127 combinations of seven heterogeneous detection tools across six score aggregation methods, producing the first (to our knowledge) empirical map of how oversight mechanisms interact in AI control ensembles. Three findings stand out: mechanism diversity matters more than tool count; k-out-of-n voting consistently dominates simpler aggregation strategies by filtering false positives through consensus; and Shapley analysis reveals that most incremental value comes from a small number of tools with genuinely independent signal - adding correlated tools hurts usefulness without improving safety.

We also introduced two novel detection mechanisms - adversarial code perturbation and paraphrase consistency checking - and demonstrated that techniques from broader AI safety research (decomposed monitoring, CoT faithfulness) can be productively adapted to the control setting. Our results provide a quantitative foundation for designing defense ensembles that balance safety, usefulness, and cost, rather than relying on theoretical arguments about which tools should complement each other.

Code and Data

- **Code repository:** <https://github.com/OrShemesh/control-arena>
- *APPS* (Hendrycks et al., 2021) via [codeparrot/apps](#) on HuggingFace; backdoor solutions from [RoganInglis/apps-control-arena](#) on HuggingFace
- *Framework:* ControlArena (Inglis et al., 2025) - <https://github.com/UKGovernmentBEIS/control-arena>

References

- Greenblatt, R., Shlegeris, B., Sachan, K., & Roger, F. (2024). AI Control: Improving Safety Despite Intentional Subversion. *Proceedings of the 41st International Conference on Machine Learning (ICML)*, PMLR 235:16295-16336. arXiv:2312.06942.
- Bhatt, A., Rushing, C., Kaufman, A., Tracy, T., Georgiev, V., Matolesi, D., Khan, A., & Shlegeris, B. (2025). Ctrl-Z: Controlling AI Agents via Resampling. arXiv:2504.10374.
- Greenblatt, R. (2025). An overview of control measures. Redwood Research Blog, April 2025. <https://blog.redwoodresearch.org/p/an-overview-of-control-measures>
- Terekhov, M., Panfilov, A., Dzenhaliou, D., Gulcehre, C., Andriushchenko, M., Prabhu, A., & Geiping, J. (2025). Adaptive Attacks on Trusted Monitors Subvert AI Control Protocols. *ICLR 2026*. arXiv:2510.09462.
- McKenzie, I.R., Hollinsworth, O.J., Tseng, T., Davies, X., Casper, S., Tucker, A.D., Kirk, R., & Gleave, A. (2025). STACK: Adversarial Attacks on LLM Safeguard Pipelines. arXiv:2506.24068.
- Bowman, S.R., et al. (2022). Measuring Progress on Scalable Oversight for Large Language Models. arXiv:2211.03540.
- Lanham, T., et al. (2023). Measuring Faithfulness in Chain-of-Thought Reasoning. arXiv:2307.13702.
- Inglis, R., Matthews, O., Tracy, T., et al. (2025). ControlArena. GitHub repository, UK AI Security Institute & Redwood Research. <https://github.com/UKGovernmentBEIS/control-arena>
- Hendrycks, D., Basart, S., Kadavath, S., et al. (2021). Measuring Coding Challenge Competence With APPS. *NeurIPS 2021 Datasets and Benchmarks Track*. arXiv:2105.09938.

Appendix

Appendix: Full Prompt Catalog

- Figure 1: *Performance vs. Number of Tools*. File: results/analysis/performance_vs_num_tools.png
- Figure 2: *Pearson Correlation Matrices between the Tools: safety (left) and usefulness (right)*. File: results/analysis/correlation_matrix.png
- Figure 3: *Safety vs Usefulness, colored by average pairwise tool correlation*. File: results/analysis/safety_usefulness_correlation.png
- Figure 4: *Average marginal contribution per tool*. File: results/analysis/marginal_contributions.png
- Figure 5: *Safety vs Usefulness - Any-k Voting*. File: results/analysis/pareto_any_k_voting.png
- Figure 6: *ROC Curves - Selected Strategies*. File: results/analysis/roc_curves.png

- All figures in this appendix were generated by `experiments/analyze_combinations.py` and `experiments/analyze_results.py` from the raw ablation scores. The raw per-sample tool scores for all samples are available at `experiments/results/ablation_scores.csv`. The full set of generated figures, intermediate analysis files, and computed operating points are available in `experiments/results/analysis/` in the project repository.

LLM Usage Statement

We used Claude for protocol design, prompt iteration, and drafting this report. Claude Code implemented the tools and ran experiments. GPT-4.1 and GPT-4.1-mini served as the untrusted and trusted models. All results were produced by automated scripts and verified against raw data.