MiniRep: ROBUST REPUTATION-BASED AGGREGATION FOR MULTI-AGENT DEBATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous agents powered by large language models (LLMs) are rapidly evolving into an open agentic ecosystem. To support trustworthy collaboration, industry initiatives increasingly assess agent reputation from past behavior and provide performance leaderboards. However, reputation derived from past performance may not reliably predict an agent's behavior on new tasks, particularly when malicious agents can adapt their behavior and influence other agents during collaboration.

We study reputation in multi-agent debate (MAD), where multiple agents answer the same query, debate to improve their answers, and aggregate them into a final output. We present MiniRep, a reputation-based aggregation system for MAD under malicious agents. To ground our threat model in established research, we construct an attack taxonomy drawing on reputation-system attacks and software-testing mutation operators, covering strategic exploitation of reputation and subtle corruption of agent proposals. Guided by this taxonomy, MiniRep evaluates agents based on both their behavior on the current task and their reputation over time, while preventing groups of agents with highly similar responses from dominating the final decision. We assess MiniRep across diverse tasks, LLM-agent compositions, corruption placements, and attack types drawn from our taxonomy. Our experimental results show that, MiniRep outperforms both conventional MAD aggregation and conventional reputation-based approaches on MATH no matter being attacked or not. Also, under a heterogeneous 10-agent setting on MATH, MiniRep outperforms all baselines in all 28 attack conditions.

1 INTRODUCTION

Autonomous agents (Wang et al., 2024; Chen et al., 2025) powered by large language models (LLMs) are evolving from standalone tools into an open agentic ecosystem. Gartner projects that the average Fortune 500 enterprise will operate more than 150,000 AI agents by 2028 (Lin, 2026). As agents are developed by different providers and increasingly interact across organizational boundaries, several industry initiatives have emerged to establish standards and infrastructure for agents to discover, connect, and interact with each other. Notable examples include the Agent2Agent (A2A) protocol (The Linux Foundation, 2026) and the ERC-8004 standard (De Rossi et al., 2025), which enable cross-provider agent interoperability and provide infrastructure for agent discovery and trust, respectively.

Along with the emergence of such open ecosystems, establishing trust among agents becomes increasingly important. Several emerging platforms therefore provide mechanisms to assess the *reputation* of LLM agents. For instance, ERC-8004 provides a Reputation Registry, a standard interface for posting and retrieving agent feedback. Several platforms and ongoing projects compute reputation scores or trust metrics from agent behavior and feedback (Credence Protocol, 2025; SAID Protocol, 2025; Chishti et al., 2026; Choe, 2025). Agent marketplaces such as AWS Marketplace (Amazon Web Services, 2026) also allow customers to rate agent products, resembling reputation mechanisms in conventional online marketplaces.

Reputation is not a new concept. Reputation systems have long been studied and deployed in peer-to-peer networks (Kamvar et al., 2003), e-commerce (Resnick & Zeckhauser, 2002), and software agents (Huynh et al., 2006; Klos & La Poutré, 2005; Mui et al., 2002). A common principle is to estimate an entity's future reliability from its past behavior or feedback. For example, Beta

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Reputation (Jøsang & Ismail, 2002) models the probability of satisfactory behavior using a Beta distribution updated from positive and negative feedback. TrueSkill (Herbrich et al., 2006) maintains a probabilistic skill rating for each player in online games and updates the ratings based on past competition outcomes. Emerging LLM agent reputation systems mentioned above largely adopt this history-based principle by deriving reputation from historical task outcomes, interactions, or feedback.

Thus, an interesting question is: *Can reputation derived from past performance reliably guide collective decisions when LLM agents may adapt their behavior or coordinate their responses?* Indeed, unlike conventional reputation settings that often involve relatively well-defined interactions, LLM agents autonomously operate on diverse and open-ended tasks, making their reliability highly task-dependent. Moreover, their autonomy, memory, tool use, and interactions with other agents introduce new attack surfaces (Zhang et al., 2025; Debenedetti et al., 2024; Kavathekar et al., 2026). These differences become particularly important when reputation determines an agent's influence on collective decisions.

In this paper, we focus on reputation at the aggregation stage of *multi-agent debate* (MAD), rather than for general-purpose LLM agents. In MAD, multiple agents independently respond to the same query, debate to improve their results, and aggregate them into a final output (Du et al., 2024; Li et al., 2024). Building on earlier work on AI debate towards safe AI (Irving et al., 2018), the problem has evolved into a general class of methods for improving LLM reasoning and factuality across diverse tasks (Du et al., 2024; Kaesberg et al., 2025; Khan et al., 2024; Choi et al., 2025). A unique feature in such a paradigm is that all the agents execute the *same task*, even when powered by different underlying LLMs. This common evaluation context makes agent behaviors directly comparable within a shared task context, providing a natural setting for studying and exploiting agent reputation.

We present MiniRep, a reputation-based aggregation system for MAD. We consider malicious agents that seek to manipulate MAD outputs away from the correct answers. Although some efforts have been made in identifying vulnerabilities of MAD, the area is relatively new, and there is no unified taxonomy of MAD vulnerabilities. We therefore construct our attack taxonomy from two complementary sources: established attacks on reputation systems (Hoffman et al., 2009), which capture strategic manipulation across agents and over time, and mutation operators from software testing (Petrović & Ivanković, 2018; Just et al., 2014), which capture subtle proposal-level corruptions. An interesting aspect is that for each attack type, we identify related mechanisms or agent behaviors reported in prior work, grounding the taxonomy in existing evidence rather than purely synthetic constructions. We summarize our taxonomy and related analogs in Table 1. Based on such a taxonomy, we then design a new way of building reputation scores for the agents, taking into consideration how agents can launch the attacks. MiniRep carefully selects a group of agents, the proposals of which are aggregated. Particular emphasis is placed on limiting the influence of agent groups with the same underlying LLM. Also, agents are rated based on both their reputation over time and behavior on the current task, such that temporal deviation can be caught. Thus, current evidence can reduce an agent's influence even after it has accumulated a strong reputation.

We assess MiniRep on three datasets, GoEmotions (Demszky et al., 2020), MATH (Hendrycks et al., 2021), and HumanEval Pro (Yu et al., 2025), under four LLM-agent compositions (i.e., using a combination of LLMs for the agents), four corruption placements (corrupting different agents), and seven attack types summarized in Table 1, yielding 112 attack conditions per dataset. We compare MiniRep with three MAD aggregation approaches (Uniform Majority (Kaesberg et al., 2025), Uniform Random-k, and Single-Metric), and four reputation systems (EigenTrust (Kamvar et al., 2003), Beta (Jøsang & Ismail, 2002), TrueSkill (Herbrich et al., 2006), and Babylon (Babylon contributors, 2026)). In these controlled aggregation runs, MiniRep has the highest observed average attacked score on three of the four task metrics, with mixed clean-task results. The most notable results are on MATH: MiniRep achieves 66.75% accuracy in clean runs, 7.50 percentage points higher than the best baseline, and 61.95% accuracy in attacked runs, compared with 54.37% for the best baseline. Across the 112 MATH attack conditions, MiniRep obtains a higher score than every baseline in 84 conditions. Also, under a heterogeneous 10-agent setting on MATH, MiniRep outperforms all baselines in all 28 attack conditions. We have made MiniRep available at `https://anonymous.4open.science/r/MiniRep-FBFE/`

Table 1: Attack taxonomy studied in this work. Reputation classes follow Hoffman et al. (2009). The last column lists related mechanisms reported in prior work; these studies do not explicitly target agent reputation. ⋆ On-off and Adaptive adapt self-promoting attacks to MAD.

Attack	Instantiation	Reputation class	Related analogue
Random coordinated	Malicious agents submit the same randomly selected incorrect answer.	Orchestrated	Adversarial peers can induce conformity and degrade agent decisions (Ko et al., 2026); incorrect proposals can propagate during MAD (Cui et al., 2026).
Optimized coordinated	Malicious agents search for an incorrect proposal that maximizes its estimated impact on the final output.	Orchestrated	Adversarial agents can generate candidate arguments and select the most persuasive one (Kraidia et al., 2026).
Diverse collusion	Malicious agents submit distinct proposals that support the same incorrect conclusion.	Orchestrated	Colluding agents can provide distinct contributions toward a common objective and induce false conclusions (Hu et al., 2026).
On-off	Malicious agents first build reputation and then attack on every subsequent task.	Self-promoting⋆	Agents can first build peer trust and later inject fabricated information (Park et al., 2026).
Adaptive	Malicious agents first build reputation. During an attack, only a selected subset deviates from the system specification.	Self-promoting⋆	Manipulating one agent can be sufficient to mislead a multi-agent system (Liu et al., 2025).
Arithmetic operator	A malicious agent changes one arithmetic operation in an honestly generated proposal.	N/A	Arithmetic and relational operator replacement can introduce small but consequential faults (Petrović & Ivanković, 2018).
Boundary value	A malicious agent modifies a boundary value or literal in an honestly generated proposal.	N/A	Literal replacement can introduce faults into otherwise unchanged programs (Just et al., 2014).

2 Problem Background and Threat Model

Multi-agent debate (MAD). Typical MAD protocols organize each task into three phases: *Propose*, in which agents independently produce candidate answers; *Debate*, which, when performed, allows agents to exchange information and revise their responses over one or more rounds; and *Decide*, in which the resulting responses are aggregated into a final output (Du et al., 2024; Chan et al., 2024). We consider a sequence of tasks indexed by t. Let $\mathcal{P} = \{1, \ldots, n\}$ denote the set of agents, and let $\tau_t = (ctx_t, \chi_t)$ denote task t, where ctx_t contains the *public evidence*, i.e., information provided to all agents, and χ_t contains task-specific parameters. In the *Propose* phase, each agent i produces a response $v_{i,t} = (c_{i,t}, \rho_{i,t})$, consisting of a payload $c_{i,t} \in \mathcal{Y}$ and an optional explanation $\rho_{i,t}$, where $\rho_{i,t} = \varnothing$ if no explanation is provided. After R debate rounds, the aggregation rule Agg first determines a pool $P_t \subseteq \mathcal{P}$ of eligible agents and then maps their responses, the task context, and any persistent state σ_{t-1} to a final response. A stateless aggregation rule simply omits σ_{t-1}.

The *Decide* phase is often the most distinctive phase of MAD protocols. Common approaches include equal-weight voting (Wang et al., 2023; Kaesberg et al., 2025; Li et al., 2024); model-based aggregation, in which an evaluator such as an LLM judge assesses the candidate responses and produces or selects the final answer (Chan et al., 2024; Hu et al., 2025); confidence-weighted voting, which gives greater influence to responses associated with higher confidence (Zheng et al., 2026); and history-based aggregation, which weights agents according to their performance on previous tasks (Ebrahimi et al., 2025). In this work, we focus on reputation-based aggregation in the *Decide* phase. We compare our approach against three aggregation baselines. Uniform Majority (Kaesberg et al., 2025) aggregates the responses of all identities using equal weights. Uniform Random-k is a size-matched control that samples k identities independently of their past performance and aggregates their responses using equal weights. Single-Metric maintains a single historical quality score for each agent and uses these scores to rank and weight the agents. Thus, Single-Metric can also be viewed as a reputation-based approach based on past performance. We formally define these baselines in Appendix A.2.

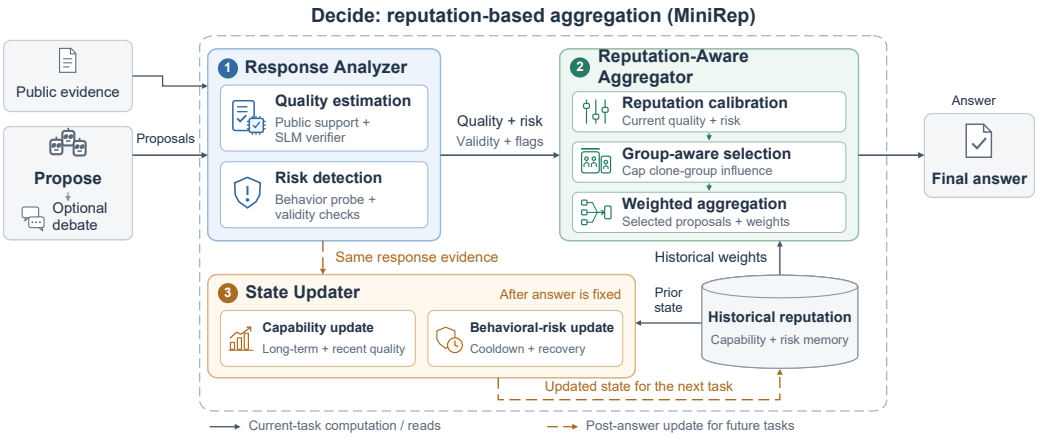

Figure 1: Overview of MiniRep.

In our system model, each agent retains a persistent identity across tasks, allowing its reputation to be updated over time. We refer to agents instantiated from the same underlying model as a *clone group*, and to the known partition of $\mathcal{P}$ into such groups as the *clone-group mapping*.

Assumptions and threat model. The adversary controls a fixed set $B \subseteq \mathcal{P}$ of at most f agents within each run. On task t, it may activate any subset $B_t \subseteq B$. Its goal is to manipulate the output of MAD, causing it to deviate from the output that would be produced if all agents behaved honestly. We assume that each malicious agent always participates in MAD and submits its candidate answer on time. A submitted proposal is observed consistently by the client and all agents. All agents also receive the same task context, including any public evidence. However, the candidate answer may deviate arbitrarily from the answer that the agent would produce if it behaved honestly. A malicious agent may behave honestly to accumulate reputation and begin attacking at any time chosen by $\mathcal{A}$. For every task, the system uses four categories of information: the proposals $\{v_i\}$, the public evidence in ctx, observable properties derived from the submitted proposals, and the clone-group mapping. The use of observable outcomes and behavior as reputation signals follows conventional reputation systems (Mui et al., 2002; Hoffman et al., 2009).

Our assumptions are practical, namely that all malicious agents submit responses and that the clone-group mapping is available to the system. First, the assumption that malicious agents always respond is reasonable within our threat model: skipping a proposal cannot directly inject adversarial content into the aggregation (Feldman et al., 2004; Hoffman et al., 2009). Second, assuming that the clone-group mapping is available is reasonable in a managed agent ecosystem, where an agent registry can expose registered identity, provider, or model information (De Rossi et al., 2025).

3 Design of MiniRep

Attack classes. As mentioned in the introduction, we construct our attack taxonomy based on established attacks on reputation systems (Hoffman et al., 2009) and mutation operators from software testing (Petrović & Ivanković, 2018; Just et al., 2014), as summarized in Table 1. These two sources capture complementary aspects of the threat model. First, because reputation determines an agent's influence on aggregation, adversaries may manipulate reputation to increase the impact of their proposals. Second, mutation operators provide systematic ways to introduce localized but consequential errors that may be difficult to detect in otherwise plausible proposals. Accordingly, we organize the attacks into three groups. First, random coordinated, optimized coordinated, and diverse collusion vary how malicious agents construct and coordinate their proposals within a task. Second, on-off and adaptive attacks manipulate reputation over time to increase adversarial influence. Finally, arithmetic-operator and boundary-value attacks apply simple mutation operators to corrupt otherwise honestly generated proposals.

Overview. We show in Fig. 1 the workflow of MiniRep. Recall that we use a reputation-based aggregation for the *Decide* phase only. Thus, we can simply follow any existing approach for the *Propose* and *Debate* phases.

Suppose that we already have a reputation system that scores each agent's capability. Let $w_{i,t}^{hist}$ denote the reputation-derived weight of agent i, and let $\mathbf{w}_t^{hist} = \left(w_{1,t}^{hist}, \ldots, w_{n,t}^{hist}\right)$ be the weights of all agents in $\mathcal{P}$. A naive reputation-based approach directly uses these historical weights to aggregate the agents' current proposals as

$$\hat{y}_t^{\text{naive}} = \mathsf{Agg}\left(\{v_{i,t}\}_{i\in\mathcal{P}}, ctx_t; \mathbf{w}_t^{hist}\right). \tag{1}$$

When conventional reputation systems rate agents based only on past performance, both self-promoting and orchestrated attacks can manipulate the aggregation output. For instance, a malicious agent with a powerful underlying LLM can stay honest to accumulate a high reputation, and then inject a wrong answer into a subsequent task to directly manipulate the output.

A running example. Consider a level-4 MATH problem asking for the center of $x^2-6x+y^2+2y = 9$ (Hendrycks et al., 2021). Completing the square gives $(x-3)^2+(y+1)^2 = 19$, so the answer is $(3,-1)$. In a 10-agent debate, suppose that three high-reputation but malicious agents submit an incorrect answer $(3,1)$ while the other seven submit $(3,-1)$. The naive weighted vote outputs $(3,1)$ if the three malicious agents carry a higher accumulated historical weight than that of the seven honest agents.

Our approach. In our work, we retain this basic idea in a *Reputation-Aware Aggregator* module, but additionally include a *Response Analyzer* that evaluates the quality of each current proposal and whether it shows signs of malicious behavior before aggregation. After the Reputation-Aware Aggregator produces the answer $\hat{y}_t$, the *State Updater* uses the signals produced by the Response Analyzer to update each agent's historical quality and behavioral-risk (i.e., previously detected suspicious behavior) records. These records determine the agents' historical weights on subsequent tasks. The three modules have distinct roles: the Response Analyzer assesses current proposals, the Reputation-Aware Aggregator adjusts their weights and limits clone-group influence, and the State Updater updates the historical reputation.

More formally, the three modules work as follows. We give the design details in Appendix B.

Response Analyzer. The Response Analyzer takes as input the proposals $v_{i,t_{i\in\mathcal{P}}}$, the public evidence ctx_t, and the clone-group mapping, and outputs quality, risk, validity, and blocking signals. Here, the public evidence refers to the information provided to all agents as part of the task, such as a question statement, a set of given facts, or a function specification.

The analyzer first evaluates each proposal. It assigns the proposal a quality score $q_{i,t}$ based on its agreement with the other proposals and an independent assessment of its semantic content. We compute a public-support score from the current proposals without using the ground-truth answer or hidden tests. This score combines agreement among proposals with the overall answer distribution. Also, we group proposals by clone groups so that proposals by agents from the same clone group are not counted as independent sources of evidence. Finally, to check the semantic content, we use a semantic verifier instantiated by a small language model (a 1.5B model in our experiments). The goal is to estimate whether the proposal is correct based on the task and the content of the proposal.

The analyzer also estimates whether the proposal exhibits malicious or manipulative behavior via a *behavior model* implemented as a multilayer perceptron (MLP) (Rumelhart et al., 1986). The behavior model checks the current proposal for signs such as instruction override and, when debate revisions are available, unexpected changes across rounds.

A proposal is blocked on the current task if no candidate answer can be extracted, or if the shared detector flags it using combined correctness and behavior signals. We show the exact rule in Appendix B.1.3.

Reputation-Aware Aggregator. The Reputation-Aware Aggregator takes the quality, risk, validity, and blocking signals produced by the Response Analyzer, together with the historical weights $w_{i,t}^{hist}{}_{i\in\mathcal{P}}$ and the clone-group mapping, and outputs an aggregated result.

For each agent, the aggregator adjusts its historical weight using the proposal's current quality relative to the other valid proposals, its current risk evidence, and the agent's recorded behavioral risk. Blocked proposals, recently suspicious agents, and responses without an extractable

answer receive additional weight penalties. Thus, a high historical weight alone does not guarantee high influence on the current task. The exact weighting and clone-group rules are deferred to Appendix B.2.1.

After that, only k out of n agents are selected into a pool $P_t \subseteq \mathcal{P}$. Before selection, the aggregator limits the total influence of each clone group so that multiple agents instantiated from the same underlying model are not treated as independent sources of reputation. It then uses the resulting weights to select the pool and determine the final aggregation weights $\{w_{i,t}\}_{i \in P_t}$. Based on the pool and the weights generated by the aggregator, the selected proposals are combined using reputation-weighted aggregation:

$$\hat{y}_t = \mathsf{Agg}(\{v_{i,t}\}_{i \in P_t}, ctx_t; \{w_{i,t}\}_{i \in P_t}) . \tag{2}$$

The task-dependent implementations of Agg are given under "Task-dependent aggregation rules" in Appendix A.2. Recall that the naive reputation-based approach is recovered by selecting all agents, $P_t = \mathcal{P}$, and directly using their historical weights, $w_{i,t} = w_{i,t}^{hist}$.

State Updater. The State Updater runs after the Reputation-Aware Aggregator produces $\hat{y}_t$. It uses the same quality, risk, validity, and blocking evidence produced by the Response Analyzer to update each agent's historical reputation for subsequent tasks.

For each agent, the updater separately records its long-term quality, recent quality, and previously detected suspicious behavior. Briefly speaking, the long-term quality and recent quality are computed based on the quality scores $q_{i,t}$ of the agent's valid proposals across previous tasks. Consistently high-quality proposals gradually improve the agent's reputation, whereas suspicious behavior reduces its influence on future tasks. Also, an unavailable or not well-formed response does not provide a new quality or risk observation.

The updated records form σ_t and determine the historical weight $w_{i,t+1}^{hist}$ used on the next task. Note that σ_t is produced only after $\hat{y}_t$ is fixed, so it cannot affect the answer to task t.

4 Experiments

4.1 Experimental Setup

Table 2: Experimental setup. Attack abbreviations follow the order in Table 1.

Name (#)	**Settings**
Dataset (3)	GoEmotions (Demszky et al., 2020), MATH (levels 4–5) (Hendrycks et al., 2021), HumanEval Pro (Yu et al., 2025)
Composition (4)	all strong, 7-strong+3-weak (7S+3W), all weak, random
Placement (4)	B (top-ranked), W (bottom-ranked), R1, R2 (random)
Attacks (7 + clean)	RAND (Hoffman et al., 2009; Ko et al., 2026), STRONG (Kraidia et al., 2026), DIV (Hu et al., 2026), ONOFF (Park et al., 2026), ADAPT (Hoffman et al., 2009; Liu et al., 2025), ARITH (Petrović & Ivanković, 2018), BOUND (Just et al., 2014)

Benchmarks. We use three datasets: GoEmotions (Demszky et al., 2020) for fine-grained multi-label emotion classification, MATH (Hendrycks et al., 2021) for competition-level mathematics (levels 4–5), and HumanEval Pro (Yu et al., 2025) for Python programming. For each dataset, we use the same fixed set of 100 tasks throughout. We provide examples from these datasets in Appendix C.

Agent pool and composition. We construct a pool of agents instantiated from seven API-accessible LLMs. Our goal is to evaluate reputation-based aggregation under meaningful differences in agent capability. We therefore select LLMs with a range of standalone scores on our datasets. This setting provides both correct and incorrect proposals for evaluating whether reputation-based aggregation assigns influence appropriately.

We rank the models from strong to weak using standalone evaluations on each dataset. The ranking is computed separately for each dataset because the relative performance of the models varies across tasks. In our experiments, we assess four compositions according to these rankings as summarized in Table 2. We provide the model rankings and the construction of each ten-agent pool in Appendix C.

Experimental setup and corruption placement. For all experiments, we fix $n = 10$, $f = 3$, and $k = 7$. We use four corruption placements: B and W compromise agents from higher- and lower-ranked models, respectively, while R1 and R2 are two random assignments.

Baselines. We assess seven baselines: three MAD aggregation baselines and four reputation systems. The MAD baselines are Uniform Majority (UMaj) (Kaesberg et al., 2025), Uniform Random-k (URand), and Single-Metric (Single). Uniform Random-k and Single-Metric are in-study controls for pool size and simple historical feedback, respectively. The reputation baselines are EigenTrust (Eigen) (Kamvar et al., 2003), Beta Reputation (Beta) (Jøsang & Ismail, 2002), TrueSkill (TS) (Herbrich et al., 2006), and Babylon (Babylon contributors, 2026). All stateful methods share post-task quality feedback; MiniRep also uses current-task signals before aggregation. We provide their definitions and adaptations in Appendix A.2.

All methods replay the same cached proposals with $R = 0$, isolating aggregation from within-task debate. Reputation still evolves across tasks, including the OnOff and Adapt schedules. We compare complete aggregation procedures, including current-task screening and the GoEmotions readout (Appendix A.2), rather than individual reputation components. Additional analyses of held-out attacks and agent compositions appear in Appendix D.1 and Appendix D.2.

Metrics. For GoEmotions (Demszky et al., 2020), we report exact-match accuracy and sample F1 (Sokolova & Lapalme, 2009). Exact-match accuracy requires the predicted and correct label sets to be identical. Sample F1 gives partial credit for overlapping labels. We use mathematical-equivalence accuracy for MATH (Hendrycks et al., 2021) and Pass@1 (Chen et al., 2021) for HumanEval Pro (Yu et al., 2025).

For each experiment, we compare a *clean* run and an *attacked* run, where the attack is launched only in the attacked run. With four corruption placements and seven attacks, we assess 28 attack conditions for each agent composition and 112 conditions per dataset. We define all the metrics we use in Appendix C.3.

4.2 Evaluation Results

We report descriptive point estimates; strict wins denote higher observed scores, not statistical significance. Additional results are given in Appendix D.

- MiniRep can improve performance even in clean runs for some tasks. The clearest gain is on MATH, where MiniRep exceeds the best baseline by 7.50 percentage points in accuracy. This is consistent with selecting and weighting proposals using both historical performance and current quality, which allows MiniRep to downweight temporarily poor responses.
- The most consistent attacked-score advantage is on MATH. MiniRep achieves the highest average attacked score on three of the four task metrics and obtains 84 strict wins across the 112 MATH attack conditions.
- Even under self-promoting attacks, i.e., OnOff and Adapt in our evaluation, where the adversary first builds reputation and then attacks, MiniRep outperforms the best baseline in five of the eight evaluated conditions and ties it in two.

Overall performance of all experiments. Table 3 summarizes average scores over the clean runs and 112 attack conditions per dataset. In clean runs, the largest gain is on MATH: MiniRep achieves 66.75% accuracy, 7.50 percentage points above the best baseline. Clean GoEmotions F1 and HumanEval Pro Pass@1 are lower than the best baselines by 2.00 and 1.75 percentage points, respectively. Under attack, MiniRep has the highest observed GoEmotions accuracy (21.16%), MATH accuracy (61.95%), and HumanEval Pro Pass@1 (78.15%), with only a 0.23-point edge on HumanEval Pro.

In Table 4(a), we fix each agent composition and summarize the number of *strict wins* among the 28 attack conditions. Our results show that MiniRep achieves broad performance gains across different agent compositions, with the most consistent improvements on MATH. Specifically, it strictly outperforms all baselines in all 28 attack conditions under the 7S+3W composition and in 27 of the 28 all-strong conditions. The gains also generally apply to other tasks: MiniRep wins 25 of the 28 7S+3W conditions for GoEmotions exact-match accuracy, 17 of the 28 all-weak conditions for GoEmotions F1, and 12 of the 28 7S+3W conditions for HumanEval Pro. These results are consis-

tent with the Reputation-Aware Aggregator using the Response Analyzer's current-quality and risk signals to adjust historical weights.

Table 3: Average task scores (%). The best evaluated baseline is selected separately for results in the clean and attacked runs. Bold marks the better score; full eight-method results are in Table 6.

	Clean		Attacked	
Dataset / metric	Best baseline	MiniRep	Best baseline	MiniRep
GoEmotions / accuracy	21.50 (Beta)	**21.75**	19.59 (Beta)	**21.16**
GoEmotions / sample F1	**36.42** (UMaj)	34.42	**34.87** (Beta)	33.94
MATH / accuracy	59.25 (UMaj)	**66.75**	54.37 (Beta)	**61.95**
HumanEval Pro / Pass@1	**79.50** (UMaj)	77.75	77.92 (Eigen)	**78.15**

Performance when capable agents are compromised. We select five attack conditions in which the compromised agents include highly ranked models, all using the 7S+3W composition. For placement, we use B for GoEmotions and R2 for MATH. We report their attacked task scores in Table 4(b). Compared with the best baseline, MiniRep achieves the highest score in four of the five conditions. In particular, it exceeds the strongest baseline by 18 and 15 percentage points under the MATH Div and Bound attacks, respectively. This result is consistent with our design: the Response Analyzer checks the current proposal, and the Reputation-Aware Aggregator adjusts its weight regardless of the agent's past reputation. The only exception is GoEmotions/Strong, where MiniRep scores 33.00% F1 and Beta scores 35.83%. This difference largely reflects their clean performance: Beta achieves 38.27% F1 and MiniRep achieves 35.00%. However, under attack, MiniRep has a smaller performance loss than Beta.

Performance under OnOff and Adapt attacks. We evaluate the OnOff and Adapt attacks under all four corruption placements using the 7S+3W composition on HumanEval Pro, i.e., eight conditions in total. Table 4(c) reports Pass@1 for all eight conditions. Our results show that MiniRep outperforms the best baseline in five out of eight conditions and ties it in two of them. The gains are 1–3 percentage points over baselines that already achieve 78%–82% Pass@1. The wins span all four corruption placements. This pattern is consistent with checking current proposals before aggregation and carrying detected risk forward to later tasks.

5 Related Work

Attacks on agents and MAD. A growing body of work documents attacks on LLM agents: indirect prompt injection hijacks an agent through the data it reads (Greshake et al., 2023; Debenedetti et al., 2024), backdoors are implanted into its memory or knowledge bases (Yang et al., 2024; Chen et al., 2024b), and social-engineering exploits propagate through agent interaction (Shapira et al., 2026; Deng et al., 2025). Multi-agent debate inherits all of these channels and adds its own: agents can be driven to conform to a wrong majority (Amayuelas et al., 2024). Some defenses assess agents within a single task. Confidence-weighted consensus probes answer confidence from prompts and hidden states to tolerate Byzantine majorities in one instance (Zheng et al., 2026). Blockchain-based coordination couples bookkeeping with multi-metric answer evaluation for one coordination run (Chen et al., 2024a). Credibility scoring also learns agent weights from past query-answering contributions (Ebrahimi et al., 2025). MiniRep combines current-proposal quality and risk assessment with persistent reputation and clone-group control.

Classical reputation does not transfer. Reputation has been studied for over two decades: electronic marketplaces aggregate transaction feedback into seller scores (Resnick & Zeckhauser, 2002), peer-to-peer systems propagate trust transitively over interaction histories (Kamvar et al., 2003), probabilistic models ground reputation in statistics (Jøsang & Ismail, 2002; Herbrich et al., 2006), and the software-agents community systematized these notions into integrated trust and reputation models (Mui et al., 2002; Huynh et al., 2006). Attacks and defenses are equally well understood: taxonomies classify attacks by the system component they target (Hoffman et al., 2009; Koutrouli & Tsalgatidou, 2012), cheap identities enable Sybil attacks in the absence of a central authority (Douceur, 2002), and TrustGuard counters strategic oscillation with PID-style update dynamics (Sri-

Table 4: Performance summary against the best evaluated baseline b. Panel (a) reports strict wins out of 28 attack conditions for each composition. Panels (b)–(c) report attacked task scores (%) where bold numbers denote the better value. The complete data are included in Tables 8–10, where repeated cells are highlighted.

(a) Strict wins under different agent compositions.

Setting	Best baseline b	MiniRep
GoEmotions Acc./7S+3W	0/28 (all)	**25/28**
GoEmotions F1 / All weak	8/28 (Single)	**17/28**
MATH / All strong	0/28 (all)	**27/28**
MATH / 7S+3W	0/28 (all)	**28/28**
HumanEval Pro / 7S+3W	10/28 (Eigen)	**12/28**

Strict task-score wins; ties are excluded.

(b) Performance under the 7S+3W composition.

Setting	Best baseline b	MiniRep
GoEmotions / B / Strong	**35.83** (Beta)	33.00
GoEmotions / B / Arith	34.00 (Beta)	**34.90**
GoEmotions / B / Bound	33.73 (Single)	**35.17**
MATH / R2 / Div	55.00 (Babylon)	**73.00**
MATH / R2 / Bound	54.00 (Babylon)	**69.00**

Attacked F1 for GoEmotions and accuracy for MATH (%).

(c) Performance under OnOff and Adapt attacks on HumanEval Pro under the 7S+3W composition.

	B/OnOff	B/Adapt	R1/OnOff	R1/Adapt	R2/OnOff	R2/Adapt	W/OnOff	W/Adapt
Baseline b	Eigen	UMaj	Eigen	Eigen	Eigen	Eigen	Babylon	Babylon
Best b	78	**81**	81	**82**	79	80	80	**81**
MiniRep	**81**	**81**	**82**	79	**82**	**83**	**81**	**81**

Attacked Pass@1 (%).

In (a), b has the largest baseline win count; in (b)–(c), it has the highest score for that condition. B, R1, R2, and W denote the four corruption placements. Attack abbreviations are listed in Table 2.

vatsa et al., 2005). These results, however, do not transfer to multi-agent debate, for two reasons. First, the feedback itself is missing: classical systems rate the observable outcome of a bilateral interaction, while in a debate the only visible outcome is the aggregated answer, so the per-agent contribution on which any reputation score must be built is never observed. Second, the attack surface differs: classical defenses assume that honest behavior eventually shows up as good outcomes, whereas debate attacks such as conformity pressure and fabricated consensus produce outcomes that look reasonable while being wrong, so the outcome signal alone cannot separate honest agents from adversarial ones. Recent reputation schemes for AI agents inherit the same outcome-based assumption (Ren et al., 2026; Lou et al., 2026; Chishti et al., 2026), and deployed registries record feedback that is rarely grounded in verifiable interactions (De Rossi et al., 2025; Xiong et al., 2026). MiniRep therefore borrows only two classical ingredients: a prior-based statistical update in the style of Jøsang & Ismail (2002) and fast reaction to recent negative behavior in the style of Srivatsa et al. (2005). It rebuilds the feedback layer itself, computing multi-signal behavioral evidence within each round and acting on it before aggregation.

6 Conclusions

We present MiniRep, a reputation-based aggregation approach for multi-agent debate (MAD). Our approach combines the quality of each agent's proposal on the current task with its historical performance. Across the evaluated aggregation settings, MiniRep achieves the highest average MATH accuracy in both clean and attacked runs. Results on the other tasks are mixed, including lower clean GoEmotions F1 and HumanEval Pro Pass@1 than the best baselines.

AI use statement

We used generative AI tools to assist with code generation and experimental results gathering. LLMs and a lightweight semantic verifier are also components of the experimental system described in Sections 3 and 4. The authors reviewed all generated materials and take responsibility for the paper, its claims, and the research artifacts.

REFERENCES

Alfonso Amayuelas, Xianjun Yang, Antonis Antoniades, Wenyue Hua, Liangming Pan, and William Yang Wang. MultiAgent collaboration attack: Investigating adversarial attacks in large language model collaborations via debate. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6929–6948, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.407. URL https://aclanthology.org/2024.findings-emnlp.407/.

Amazon Web Services. Ai agents and tools in aws marketplace. https://aws.amazon.com/marketplace/solutions/ai-agents-and-tools, 2026. Accessed: 2026-09-25.

Babylon contributors. Babylon: Reputation calculation service. Official software repository, file reputation-calculation-service.ts, 2026. URL https://github.com/BabylonSocial/babylon/blob/9b18096e3e69e68055ea4fc8cb0a1a94f80c2cef/packages/engine/src/reputation/reputation-calculation-service.ts. Accessed 2026-09-19. The cited source defines the composite score; the task-feedback adapter is specified in this paper.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9079–9093, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25cc3adf8c85f7c70989cb8a97a691a7-Abstract-Conference.html.

Bei Chen, Gaolei Li, Xi Lin, Zheng Wang, and Jianhua Li. Blockagents: Towards byzantine-robust llm-based multi-agent coordination via blockchain. In *Proceedings of the ACM Turing Award Celebration Conference - China 2024*, ACM-TURC '24, pp. 187–192, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400710117. doi: 10.1145/3674399.3674445. URL https://doi.org/10.1145/3674399.3674445.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 36374–36411, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/59c27bf8d56d3d50c7aeaf7535dee975-Abstract-Conference.html.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 130185–130213. Curran Associates, Inc., 2024b. doi: 10.52202/079017-4136. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb113910e9c3f6242541c1652e30dfd6-Abstract-Conference.html.

Mohd Sameen Chishti, Damilare Peter Oyinloye, and Jingyue Li. Agentreputation: A decentralized agentic ai reputation framework. In *Proceedings of the 34th ACM International Conference on the Foundations of Software Engineering*, pp. 1317–1321, 2026. doi: 10.1145/3803437.3805579. URL https://doi.org/10.1145/3803437.3805579.

Hanjoon Choe. agent-reputation-sdk: Ethereum SDK extensions for ERC-8004 trustless agents. https://github.com/hanjoonchoe/agent-reputation-sdk, 2025. Software repository. Accessed: 2026-09-25.

Hyeong Kyu Choi, Xiaojin Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3405. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/934252acd87f254d5d4672fbde283bd2-Abstract-Conference.html.

Credence Protocol. Credence: Real-time reputation scoring for AI agents. https://www.credenceprotocol.com/, 2025. Accessed: 2026-09-25.

Yu Cui, Hang Fu, Haibin Zhang, Licheng Wang, and Cong Zuo. Free-mad: Consensus-free multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 31977–31997, 2026. doi: 10.18653/v1/2026.findings-acl.1600. URL https://aclanthology.org/2026.findings-acl.1600/.

Marco De Rossi, Davide Crapis, Jordan Ellis, and Erik Reppel. Ethereum improvement proposals. erc-8004: Trustless agents. https://eips.ethereum.org/EIPS/eip-8004, 2025. Draft specification. Accessed: 2026-09-25.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 82895–82920. Curran Associates, Inc., 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4040–4054. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.372. URL https://aclanthology.org/2020.acl-main.372/.

Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *ACM Comput. Surv.*, 57(7), February 2025. ISSN 0360-0300. doi: 10.1145/3716628. URL https://doi.org/10.1145/3716628.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

John R. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron (eds.), *Peer-to-Peer Systems*, pp. 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45748-0. URL https://www.microsoft.com/en-us/research/publication/the-sybil-attack/.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Sana Ebrahimi, Mohsen Dehghankar, and Abolfazl Asudeh. An adversary-resistant multi-agent LLM system via credibility scoring. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1676–1693, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.90. URL https://aclanthology.org/2025.ijcnlp-long.90/.

Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, EC '04, pp. 102–111, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581137710. doi: 10.1145/988772.988788. URL https://doi.org/10.1145/988772.988788.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, pp. 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems*, volume 19, pp. 569–576, 2006. URL https://proceedings.neurips.cc/paper/2006/hash/f44ee263952e65b3610b8ba51229d1f9-Abstract.html.

Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1–31, 2009. doi: 10.1145/1592451.1592452. URL https://doi.org/10.1145/1592451.1592452.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jinwei Hu, Xinmiao Huang, Youcheng Sun, Yi Dong, and Xiaowei Huang. Lying with truths: Open-channel multi-agent collusion for belief manipulation via generative montage. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5979–5996, 2026. doi: 10.18653/v1/2026.acl-long.270. URL https://aclanthology.org/2026.acl-long.270/.

Tianyu Hu, Zhen Tan, Song Wang, Huaizhi Qu, and Tianlong Chen. Multi-agent debate for llm judges with adaptive stability detection. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 46504–46540. Curran Associates, Inc., 2025. doi: 10.52202/085713-1548. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/42475c537936b2394b5015e871765056-Abstract-Conference.html.

Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Autonomous agents and multi-agent systems*, 13(2): 119–154, 2006. doi: 10.1007/s10458-005-6825-4. URL https://link.springer.com/article/10.1007/s10458-005-6825-4.

Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018. URL https://arxiv.org/abs/1805.00899.

Audun Jøsang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002. URL https://aisel.aisnet.org/bled2002/41/.

René Just, Darioush Jalali, Laura Inozemtseva, Michael D Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing? In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pp. 654–665, 2014. doi: 10.1145/2635868.2635929. URL https://doi.org/10.1145/2635868.2635929.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11640–11671, 2025. doi: 10.18653/v1/2025.findings-acl.606. URL https://aclanthology.org/2025.findings-acl.606/.

Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web*, pp. 640–651. ACM, 2003. doi: 10.1145/775152.775242. URL https://dl.acm.org/doi/10.1145/775152.775242.

Archit Karandikar, Nicholas Cain, Dustin Tran, Balaji Lakshminarayanan, Jonathon Shlens, Michael Mozer, and Becca Roelofs. Soft calibration objectives for neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/f8905bd3df64ace64a68e154ba72f24c-Abstract.html.

Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. TAMAS: Benchmarking adversarial risks in multi-agent LLM systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31238–31268, 2026. doi: 10.18653/v1/2026.acl-long.1442. URL https://aclanthology.org/2026.acl-long.1442/.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive LLMs leads to more truthful answers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23662–23733. PMLR, 2024. URL https://proceedings.mlr.press/v235/khan24a.html.

Tomas B Klos and Han La Poutré. Decentralized reputation-based trust for assessing agent reliability under aggregate feedback. In *Trusting Agents for Trusting Electronic Societies*, volume 3577 of *Lecture Notes in Computer Science*, pp. 110–128. Springer, 2005. doi: 10.1007/11532095_7. URL https://link.springer.com/chapter/10.1007/11532095_7.

Changgeon Ko, Jisu Shin, Hoyun Song, Huije Lee, Eui Jun Hwang, and Jong C Park. Social dynamics as critical vulnerabilities that undermine objective decision-making in llm collectives. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37865–37890, 2026. doi: 10.18653/v1/2026.acl-long.1756. URL https://aclanthology.org/2026.acl-long.1756/.

Eleni Koutrouli and Aphrodite Tsalgatidou. Taxonomy of attacks and defense mechanisms in p2p reputation systems—lessons for reputation system designers. *Computer Science Review*, 6(2–3):47–70, 2012. ISSN 1574-0137. doi: 10.1016/j.cosrev.2012.01.002. URL https://www.sciencedirect.com/science/article/pii/S1574013712000093.

Insaf Kraidia, Iyas Qaddara, Alhanof Almutairi, Nada Alzaben, and Samir Brahim Belhouari. When collaboration fails: persuasion driven adversarial influence in multi agent large language model debate. *Scientific Reports*, 16(1):11640, 2026. doi: 10.1038/s41598-026-42705-7. URL https://www.nature.com/articles/s41598-026-42705-7.

Heungsub Lee. TrueSkill: Python implementation documentation. Software documentation, version 0.4.5, 2026. URL https://trueskill.org/. Accessed September 25, 2026.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=bgzUSZ8aeg.

Belle Lin. Companies have a new ai problem: Too many agents. *The Wall Street Journal*, May 2026. URL https://www.wsj.com/cio-journal/companies-have-a-new-ai-problem-too-many-agents-9539c4d6. Accessed: September 17, 2026.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017. doi: 10.1109/ICCV.2017.324. URL https://doi.org/10.1109/ICCV.2017.324.

Fengyuan Liu, Rui Zhao, Shuo Chen, Guohao Li, Philip Torr, Lei Han, and Jindong Gu. Can an individual manipulate the collective decisions of multi-agents? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12158–12182, 2025. doi: 10.18653/v1/2025.emnlp-main.611. URL https://aclanthology.org/2025.emnlp-main.611/.

Yuwei Lou, Hao Hu, Shaocong Ma, Zongfei Zhang, Liang Wang, Jidong Ge, and Xianping Tao. DRF: LLM-AGENT dynamic reputation filtering framework. In *Neural Information Processing*, volume 16312 of *Lecture Notes in Computer Science*, pp. 127–141. Springer, 2026. doi: 10.1007/978-981-95-4384-7_10. URL https://link.springer.com/chapter/10.1007/978-981-95-4384-7_10.

Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pp. 280–287, 2002. doi: 10.1145/544741.544807. URL https://doi.org/10.1145/544741.544807.

Jeongsu Park, Yuji Lim, Geonwoo Kim, Taehyeon Yun, and Moohong Min. Data poisoning in llm multiagent societies: Social proof drives collective decision failure in financial deliberation. *ETRI Journal*, pp. 1–13, 2026. doi: 10.4218/etrij.2026-0186. URL https://onlinelibrary.wiley.com/doi/10.4218/etrij.2026-0186.

Goran Petrović and Marko Ivanković. State of mutation testing at google. In *Proceedings of the 40th international conference on software engineering: Software engineering in practice*, pp. 163–171, 2018. URL https://research.google/pubs/state-of-mutation-testing-at-google/.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115. Version 2.

Siyue Ren, Wanli Fu, Xinkun Zou, Chen Shen, Yi Cai, Chen Chu, Zhen Wang, and Shuyue Hu. Reputation as a solution to cooperation collapse in LLM-based MASs. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, pp. 245–253. International Foundation for Autonomous Agents and Multiagent Systems, 2026. doi: 10.65109/UEHN4980. URL https://doi.org/10.65109/UEHN4980.

Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. In Michael R. Baye (ed.), *The Economics of the Internet and E-commerce*, pp. 127–157. Emerald Group Publishing Limited, 10 2002. ISBN 978-0-76230-971-9. doi: 10.1016/S0278-0984(02)11030-3. URL https://doi.org/10.1016/S0278-0984(02)11030-3.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0.

SAID Protocol. SAID: The identity and reputation layer for AI agents. https://www.saidprotocol.com/, 2025. Accessed: 2026-09-25.

Natalie Shapira, Chris Wendler, Avery Yen, Gabriele Sarti, Koyena Pal, Olivia Floody, Adam Belfki, Alex Loftus, Aditya Ratan Jannali, Nikhil Prakash, et al. Agents of chaos. *arXiv preprint arXiv:2602.20021*, 2026. URL https://arxiv.org/abs/2602.20021.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. doi: 10.1016/j.ipm.2009.03.002. URL https://www.sciencedirect.com/science/article/pii/S0306457309000259.

Mudhakar Srivatsa, Li Xiong, and Ling Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pp. 422–431, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930469. doi: 10.1145/1060745.1060808. URL https://doi.org/10.1145/1060745.1060808.

The Linux Foundation. Agent2agent (a2a) protocol. https://a2a-protocol.org/latest/, 2026. Accessed: 2026-09-25.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of computer science*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1. URL https://link.springer.com/article/10.1007/s11704-024-40231-1.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Xihan Xiong, Zelin Li, Wei Wei, Qin Wang, William Knottenbelt, and Zhipeng Wang. Can trustless agents be trusted? an empirical study of the ERC-8004 decentralized AI agent ecosystem. *arXiv preprint arXiv:2606.26028*, 2026. URL https://arxiv.org/abs/2606.26028.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to llm-based agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 100938–100964. Curran Associates, Inc., 2024. doi: 10.52202/079017-3201. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b6e9d6f4f3428cd5f3f9e9bbae2cab10-Abstract-Conference.html.

Zhaojian Yu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. HumanEval Pro and MBPP Pro: Evaluating large language models on self-invoking code generation task. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13253–13279. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.686. URL https://aclanthology.org/2025.findings-acl.686/.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. In *International Conference on Learning Representations*, volume 2025, pp. 35331–35366, 2025. URL https://openreview.net/forum?id=V4y0CpX4hK.

Lifan Zheng, Jiawei Chen, Qinghong Yin, Jingyuan Zhang, Xinyi Zeng, and Yu Tian. Rethinking the reliability of multi-agent system: A perspective from Byzantine fault tolerance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35012–35020, 2026. doi: 10.1609/aaai.v40i41.40806. URL https://ojs.aaai.org/index.php/AAAI/article/view/40806.

A NOTATIONS AND BASELINES

A.1 NOTATIONS

Table 5 collects the symbols used in Sec. 2, Sec. 3, and the detailed definitions below. Identities are indexed by i and tasks by t. Per-task quantities carry a task subscript only where the task matters. Task t reads σ_{t-1} and writes σ_t after the answer; method-specific quantities use local superscripts where needed.

Table 5: Core notation.

Symbol	Meaning
Protocol related	
$\mathcal{P}$; n; f	Identity set; number of identities; maximum adversarial identities.
$\tau = (ctx, \chi)$	Task: context ctx (statement and public evidence) and parameter χ.
$v_i = (c_i, \rho_i)$	Response of identity i: payload c_i and rationale ρ_i.
R	Debate rounds ($R = 0$ in the main setting).
G; clone-group mapping	Known group of identities; the experiments group replicas by their underlying API model.
$\mathcal{A}$; B_t	Adversary operator; coalition active on task t, $\lvert B_t\rvert \le f$.
P_t; k	Identities selected on a task; target pool size.
Agg; σ	Aggregation rule (the *Decide* step); persistent reputation state.
Same-task signals	
$p^{pub}_{i,t}$	Public support (Eq. 32); distinct from the task context ctx.
$p^{sem}_{i,t}$	Calibrated correctness probability from the 1.5B semantic verifier.
$r^{atk}_{i,t}$	Behavior-model risk score (Appendix B.1.2).
$\eta_{i,t}$; $f_{i,t}$	Parse-validity indicator; shared detector flag. The response remains $v_{i,t}$.
$a_{i,t}$; $r^{gate}_{i,t}$	Fused evidence score; derived decision penalty, not a second learned probability (Eq. 41).
$\widehat{a}_{i,t}$	Alarm strength used in persistent updates: $\max\{a_{i,t}, 0.90 d_{i,t}\}$.
q^{med}_t; $s^{self}_{i,t}$	Median parse-valid quality; same-model self-consistency.
$c^{fmt}_{i,t}$	Format compliance in $[0, 1]$; parse failure forces $q_{i,t} = 0$.
$s_{i,t}$; $q_{i,t}$	Fused support; same-task quality (Appendix B.1.3).
$d_{i,t}$	Current-task block: parse failure or the shared detector flag $f_{i,t}$ (Eq. 39).
blocked set	Current-task blocked identities or identities with active temporary restrictions. They rank behind all unblocked candidates.
Weights and selection	
$w^{hist}_{i,t}$	Incoming weight from σ_{t-1}, with numerical floor (Eq. 51).
$w_{i,t}$; $Q_{i,t}$	Final pool weight; historical-risk/cooldown mark used in current weighting.
$\widetilde{w}_{i,t}$; $m_{i,t}$	Pre-group current-round weight (Eq. 42); product of blocking, historical-risk and invalidity multipliers.
M_G; p_G	Pre-group weight sum; largest member weight in G, with $m = \lvert G\rvert$.
Persistent state	
$\bar{q}_i$; q^{short}_i	Long-term quality (Beta-style prior, strength 12); short-term quality (asymmetric EMA $0.30/0.08$).
$\bar{e}_i$; e^{short}_i	Long- and short-term correctness support.
D_i; K_i; E_i; I_i	Failure debt; strike counter; detector EMA; identity risk (Appendix B.3.1).
r^{temp}_i	Temporal risk: maximum of D_i, level risk, E_i, I_i, and instant failure risk.
h_i	Cooldown counter (three tasks after a valid-response detection).
Baseline and evaluation quantities	
$y^{cand}_{i,t}$; F_t; $\mathcal{F}_t$	Parsed candidate answer; feedback-available identities; detector-selected candidates.
$z_{i,t}$; $u_{i,t}$; $x_{i,t}$	Estimated feedback quality; validity-masked score; format-adjusted score.
$\sigma^{\mathrm{SM}}_{i,t}$; h^{TS}_i	Single-Metric quality EMA; TrueSkill conservative rating.
$S^{\mathrm{Bab}}_{i,t}$; $B_{i,t}$	Babylon score; earlier-performance baseline for gap tracking (Appendix B.3.2).
A_t; J_t; E_t; H_t	Active-attack indicator; payload-bearing identities; exposure and payload-match indicators.
$s^0_{m,t}$; $s^a_{m,t}$	Clean and attacked per-task utility for method m.

A.2 BASELINES

We distinguish source-method defaults from settings introduced by our MAD adaptations; feedback mappings, clipping bounds, and numerical tolerances are implementation choices unless stated otherwise.

Multi-agent debate, formally. We formalize the MAD phases introduced in Sec. 2, using the notation in Table 5. Consider a sequence of tasks $\tau_t = (ctx_t, \chi_t)$ indexed by $t = 1, 2, \ldots$. In the *Propose* phase, each identity $i \in \mathcal{P}$ independently generates

$$v_{i,t}^{(0)} = \mathrm{Propose}_i(\tau_t). \tag{3}$$

If the protocol includes debate, then at each round $\ell = 1, \ldots, R$, identity i updates its proposal according to

$$v_{i,t}^{(\ell)} = \mathrm{Debate}_i\left(\tau_t, v_{i,t}^{(\ell-1)}, \mathcal{H}_{i,t}^{(\ell-1)}\right), \tag{4}$$

where $\mathcal{H}_{i,t}^{(\ell-1)}$ contains the proposals and messages visible to identity i before round ℓ. The MAD communication topology determines the contents of this history. We use

$$v_{i,t} = v_{i,t}^{(R)} = (c_{i,t}, \rho_{i,t}) \tag{5}$$

to denote the final proposal submitted to the *Decide* phase. When $R = 0$, as in our main setting, this is simply the initial proposal $v_{i,t}^{(0)}$.

The *Decide* phase applies an aggregation rule Agg to a selected pool $P_t \subseteq \mathcal{P}$. For the voting-based methods considered below, let $y_{i,t}^{cand}$ denote the normalized answer parsed from the payload $c_{i,t}$. We set $y_{i,t}^{cand} = \perp$ if parsing fails. Given nonnegative weights $w_{i,t}$ for the selected identities, the final answer is

$$\hat{y}_t \in \arg\max_{a \neq \perp} \sum_{i \in P_t} w_{i,t} \mathbb{I}[y_{i,t}^{cand} = a]. \tag{6}$$

Eq. 6 gives weighted answer-group voting for MATH, with equality interpreted through the protocol's candidate comparison. Label-set and code readouts are specified under task-dependent aggregation below.

To compare aggregation methods independently of LLM-generation variance, we generate and cache the final proposal set

$$\mathcal{R}_t = \{v_{i,t} : i \in \mathcal{P}\} \tag{7}$$

once for each task. Every aggregation method receives the same $\mathcal{R}_t$. Thus, any difference in the resulting decision is caused by which agents are included in P_t, how much weight is assigned to each selected proposal, and how the weighted proposals are combined into the final answer, rather than by different sampled agent responses.

Estimated-quality feedback. To support the stateful methods evaluated in Sec. 4.1, we define an estimated-quality feedback interface. This interface is introduced for our experimental comparison and is not a standard component of MAD protocols. For each identity i whose proposal is observed on task t, let $z_{i,t} \in [0, 1]$ denote an estimated quality score, $\eta_{i,t} \in \{0, 1\}$ indicate whether the proposal can be parsed successfully, and $c_{i,t}^{fmt} \in [0, 1]$ measure format compliance. We define

$$u_{i,t} = \eta_{i,t} z_{i,t}, \qquad x_{i,t} = z_{i,t}(0.75 + 0.25 c_{i,t}^{fmt}).$$

These quantities provide estimated-quality signals rather than benchmark ground-truth labels. Stateful aggregation methods update their persistent state only after the decision on task t has been made.

Uniform Majority and Uniform Random-k. Neither Uniform Majority nor Uniform Random-k maintains persistent state. Uniform Majority applies equal-weight voting to all identities (Kaesberg et al., 2025):

$$P_t = \mathcal{P}, \qquad w_{i,t} = 1 \quad (i \in \mathcal{P}). \tag{8}$$

Its pool therefore contains all $n = 10$ identities. This baseline shows the result when every agent participates and has the same influence, regardless of its performance on previous tasks.

Uniform Random-k is our size-matched sampling control. It samples a size-k pool uniformly without replacement and assigns equal weight to every selected identity:

$$\Pr(P_t = S) = \binom{n}{k}^{-1} \quad \text{for every } S \subseteq \mathcal{P} \text{ with } |S| = k, \qquad w_{i,t} = 1 \quad (i \in P_t). \tag{9}$$

The selection is independent of σ, agent feedback, and past performance. To make the control reproducible, the implementation uses a fixed pseudorandom seed for each experimental run and combines it with the task identifier when sampling P_t. Repeating the same run therefore yields the same pool for every task. This baseline isolates the effect of using only k identities without using reputation for selection or weighting.

For the stateful top-k methods considered below, P_t instead contains the k identities with the largest weights available before task t. Weight ties are resolved using a deterministic seeded rule. During the configured all-identity warm-up period, these baselines use $P_t = \mathcal{P}$. In contrast, MiniRep applies its same-task screening mechanism and constructs a size-k pool even during warm-up.

Single-Metric. Single-Metric is an in-study control designed to test what can be achieved using only one historical quality indicator per identity. It excludes the additional mechanisms used by MiniRep, including identity-risk detection, clone-group caps, cooldown, and same-task screening.

Let $\sigma^{\mathrm{SM}}_{i,t}$ denote the exponentially smoothed historical quality of identity i after incorporating feedback available through task t. This symbol is distinct from the fused same-task support $s_{i,t}$ defined in Table 5. We initialize

$$\sigma^{\mathrm{SM}}_{i,0} = 0.5 \quad (i \in \mathcal{P}). \tag{10}$$

Before task t, Single-Metric uses the state available through task $t-1$ to assign

$$w_{i,t} = \mathrm{clip}_{[0.05,\,1.50]}\left(\sigma^{\mathrm{SM}}_{i,t-1}\right). \tag{11}$$

It selects the k identities with the largest $w_{i,t}$ values and applies the weighted-vote rule in Eq. 6. The same scalar historical score therefore controls both agent selection and voting weight.

After feedback for task t becomes available, the score of every observed identity is updated using a fixed-rate exponential moving average. Let F_t denote the identities for which feedback is available for this update (distinct from the detector's candidate set $\mathcal{F}_t$):

$$\sigma^{\mathrm{SM}}_{i,t} = 0.85\sigma^{\mathrm{SM}}_{i,t-1} + 0.15x_{i,t} \quad (i \in F_t). \tag{12}$$

For an unobserved identity, the previous score is carried forward:

$$\sigma^{\mathrm{SM}}_{i,t} = \sigma^{\mathrm{SM}}_{i,t-1} \quad (i \notin F_t). \tag{13}$$

The update coefficient 0.15 controls how quickly recent feedback changes the historical estimate. Single-Metric otherwise maintains no additional persistent signals or current-task controls. Its initialization, clipping interval, and update rate are fixed settings of this in-study control.

The four reputation baselines below receive the same quality feedback $u_{i,t}$ after task t. Each baseline uses this feedback to update the weight $w_{i,t+1}$ for the next task. Unlike MiniRep, these baselines do not use the current proposal to adjust an agent's weight before the current answer is produced.

Babylon. We adapt Babylon (Babylon contributors, 2026) to maintain one reputation score for each agent. Let n_i be the number of observed tasks, and let

$$W_i = \frac{1}{n_i}\sum_{\tau} \mathbf{1}[z_{i,\tau} \geq 0.999] \tag{14}$$

be the fraction of observations with a nearly perfect verifier score. The adaptation also maintains F_i and J_i as running averages of

$$(20 + 80z_{i,t})\left(0.75 + 0.25c^{fmt}_{i,t}\right). \tag{15}$$

Thus, both records increase with the verifier score and with compliance with the required output format. A separate record $V_{i,t}$ tracks the verifier score over time:

$$V_{i,0} = 0.5, \qquad V_{i,t} = 0.90V_{i,t-1} + \frac{0.10}{1 + \exp[-(2z_{i,t} - 1)]}. \tag{16}$$

Babylon combines these records with the number of observed tasks:

$$\begin{aligned} S_{i,t}^{\mathrm{Bab}} &= 0.4\,[100\,(0.7V_{i,t} + 0.3W_i)] + 0.4\,(0.7F_i + 0.3J_i) \\ &\quad + 0.2\min\{100, 2n_i\}, \\ w_{i,t+1} &= \max\{0.05, S_{i,t}^{\mathrm{Bab}}/100\}. \end{aligned} \tag{17}$$

The initial reputation is 50. The $0.4/0.4/0.2$ component weights, $0.7/0.3$ blends, activity term, and initial score follow the cited implementation; the verifier-based feedback mapping is our MAD adaptation. This baseline evaluates whether a single score combining past performance, format compliance, and the number of observed tasks is sufficient for MAD aggregation.

Beta Reputation. Beta Reputation (Jøsang & Ismail, 2002) accumulates positive and negative evidence for each agent. The quality feedback $u_{i,t}$ is treated as fractional positive evidence, and $1-u_{i,t}$ is treated as fractional negative evidence:

$$\alpha_{i,0} = \beta_{i,0} = 1, \qquad \alpha_{i,t} = \alpha_{i,t-1} + u_{i,t}, \qquad \beta_{i,t} = \beta_{i,t-1} + 1 - u_{i,t}. \tag{18}$$

The weight for the next task is proportional to the estimated fraction of positive evidence:

$$w_{i,t+1} \propto \frac{\alpha_{i,t}}{\alpha_{i,t} + \beta_{i,t}}. \tag{19}$$

The weights are normalized across agents. All previous observations remain in α_i and β_i, so older evidence is not discarded or reduced. The $(1, 1)$ initialization corresponds to a uniform Beta prior. This baseline evaluates direct accumulation of quality evidence without maintaining a separate record of suspicious behavior.

EigenTrust. EigenTrust (Kamvar et al., 2003) originally computes trust from ratings between participants. Our setting does not collect pairwise ratings from agents. We therefore use the shared quality feedback $u_{j,t}$ as evidence about each observed agent j. For every pair (i, j), we update

$$S_{ij}^{+} \leftarrow S_{ij}^{+} + u_{j,t}, \qquad S_{ij}^{-} \leftarrow S_{ij}^{-} + 1 - u_{j,t}, \tag{20}$$

with $S_{ij}^{+} = S_{ij}^{-} = 0$ initially. Because the same feedback is available to the system, every row receives the same evidence about agent j. We define

$$s_{ij} = \max\{S_{ij}^{+} - S_{ij}^{-}, 0\}, \tag{21}$$

and normalize each row:

$$C_{ij} = \begin{cases} s_{ij} / \sum_{\ell} s_{i\ell}, & \sum_{\ell} s_{i\ell} > 0, \\ 1/n, & \text{otherwise.} \end{cases} \tag{22}$$

The weights are obtained by repeatedly applying

$$\boldsymbol{w} = 0.85 C^{\mathsf{T}} \boldsymbol{w} + 0.15 \boldsymbol{p}, \qquad p_i = 1/n. \tag{23}$$

The computation starts from $\boldsymbol{p}$ and stops when the ℓ_1 change is at most 10^{-10} or after 100 iterations. The 0.15 damping, uniform prior, and stopping rule are fixed settings of this adaptation. This adaptation applies EigenTrust's repeated trust calculation to the feedback available in our setting. It does not assume that agents provide independent ratings of one another.

TrueSkill. TrueSkill (Herbrich et al., 2006) maintains a skill estimate and its uncertainty for each agent:

$$\theta_i \sim \mathcal{N}(\mu_i, \sigma_i^2). \tag{24}$$

After each task, agents are ranked according to decreasing $u_{i,t}$. Scores that differ by at most 10^{-12} are treated as tied. TrueSkill updates μ_i and σ_i from these rankings. We use the initialization and update defaults of the Python `trueskill` implementation (Lee, 2026):

$$\mu_0 = 25, \qquad \sigma_0 = 25/3, \qquad \beta_{\mathrm{TS}} = 25/6, \qquad \tau = 25/300, \qquad p_{\mathrm{draw}} = 0.10. \tag{25}$$

The score

$$h_i^{\mathrm{TS}} = \mu_i - 3\sigma_i \tag{26}$$

favors agents with high estimated skill and low uncertainty. We convert it into a positive weight for the next task:

$$w_{i,t+1} \propto \exp\left(\mathrm{clip}_{[-50,50]}\,\frac{h_i^{\mathrm{TS}} - \min_j h_j^{\mathrm{TS}}}{\max\{1, \beta_{\mathrm{TS}}\}}\right). \tag{27}$$

This baseline evaluates whether accounting for uncertainty in an agent's estimated capability improves aggregation. It does not maintain a separate record of suspicious behavior.

Task-dependent aggregation rules. These rules implement Agg in Eq. 2, corresponding to weighted aggregation in Fig. 1. After each method determines the aggregation pool P_t and weights $\{w_{i,t}\}_{i\in P_t}$, a task-dependent aggregation rule converts the selected proposals into the final answer $\hat{y}_t$. These rules are used by all evaluated methods. The MATH and HumanEval Pro rules are identical across methods. For GoEmotions, the seven baselines share one rule, while MiniRep uses the alternative rule defined below. The task-specific scoring coefficients, label thresholds, and label-count cap are fixed choices of these aggregation implementations, not benchmark-prescribed defaults.

Let $V_t \subseteq P_t$ contain the agents whose proposals provide valid, nonempty candidate answers, and let $W_t = \sum_{i\in V_t} w_{i,t}$. Proposals without a valid candidate answer receive no aggregation weight. If V_t is empty, the method returns an empty answer, which is evaluated as incorrect.

For MATH, we group candidate answers according to mathematical equivalence and sum the weights within each group. The group with the largest total weight determines the final answer. This comparison uses only the submitted candidate answers and does not use the reference answer.

For HumanEval Pro, the aggregation rule selects the candidate program $i \in V_t$ with the highest score:

$$2p_i^{\mathrm{example}} + 0.9\frac{\sum_{j\in V_t:\, F_j=F_i} w_{j,t}}{W_t} + 0.35\frac{w_{i,t}}{W_t} + 0.25 s_i^{\mathrm{static}}, \tag{28}$$

where p_i^{example} measures whether the program passes the examples included in the task, F_i identifies programs with the same normalized abstract syntax tree, and s_i^{static} checks whether the program is syntactically valid and contains the required entry point. The hidden evaluation tests are not used during aggregation.

For GoEmotions, the seven baselines distribute each proposal's weight uniformly across its predicted label set Y_i:

$$A_\ell = \sum_{i\in V_t} w_{i,t}\frac{\mathbf{1}[\ell \in Y_i]}{|Y_i|}. \tag{29}$$

They return at most five labels satisfying

$$A_\ell \geq \max\left\{0.45\max_h A_h,\ 0.18 W_t\right\}, \tag{30}$$

and return the highest-scoring label if no label satisfies this condition. In contrast, MiniRep selects one of the label sets proposed by the agents. It chooses the set Y that maximizes

$$0.50\frac{\sum_{i\in V_t:Y_i=Y} w_{i,t}}{W_t} + 0.40\frac{\sum_{i\in V_t} w_{i,t}\,\mathrm{F1}(Y,Y_i)}{W_t} + 0.10\frac{|\{g(i) : i\in V_t,\ Y_i = Y\}|}{|\{g(i) : i\in V_t\}|}, \tag{31}$$

where $g(i)$ denotes the clone group of agent i. This rule considers the total weight supporting Y, its agreement with the other proposed label sets, and the number of clone groups that propose it. Therefore, the GoEmotions experiments compare the complete aggregation procedures rather than isolating the effect of reputation weights alone.

B Deferred Design Details of MiniRep

The following details are organized by the three modules in Fig. 1 and Sec. 3.

Unless attributed to a cited method or checkpoint calibration, the numerical settings below are fixed choices of our implementation.

B.1 Response Analyzer

B.1.1 Public-evidence Score

This is the public-support score used by the Response Analyzer in Sec. 3. Each candidate receives a score computed from the cohort alone, with no gold labels and no hidden tests.

$$p_{i,t}^{pub} = \eta_{i,t}\big(\beta_1\pi_{i,t} + \beta_2\gamma_{i,t} + \beta_3\lambda_{i,t} + \beta_4 s_{i,t}^{self} + \beta_5(1-\varepsilon_t)\big), \tag{32}$$

Here π is cohort-agreement probability, γ cross-model agreement, and λ leave-one-group-out plurality agreement. The remaining terms are same-model self-consistency s^{self} and normalized answer-distribution entropy ε_t. We fix the implementation weights at $\beta = (0.45, 0.25, 0.15, 0.10, 0.05)$, and $\eta_{i,t}$ is the parse-validity indicator. We compute all five statistics after grouping identities by underlying model, so replicas of one model count as one evidence source.

B.1.2 Local feedback models

For the semantic verifier in Sec. 3, we fine-tune `Qwen2.5-1.5B-Instruct` as a binary correctness classifier (Qwen et al., 2025). Its input contains the task with gold fields removed, the candidate response, parse validity, answer-group sizes, same-model consistency, cohort agreement, and normalized cohort entropy. Agent and model identifiers, attack names, Byzantine labels, gold answers, and hidden tests are excluded. We use 4-bit LoRA (Hu et al., 2022; Dettmers et al., 2023) ($r = 16$, $\alpha = 32$) with class-weighted focal cross-entropy (Lin et al., 2017) and a soft-ECE regularizer (Karandikar et al., 2021), followed by temperature scaling (Guo et al., 2017). The rank and scaling factor are fixed fine-tuning settings of our verifier, rather than defaults prescribed by these methods.

$$\mathcal{L}_{\text{ver}} = \frac{1}{N}\sum_j (1-p_{y_j})^{\gamma}\,\text{CE}(z_j, y_j) + \lambda_{\text{cal}}\widetilde{\text{ECE}}, \tag{33}$$

Splits are made by canonical task, which prevents attack variants of one task from crossing partitions. The calibrated output is $p_{i,t}^{sem}$. The attack families used to train the local feedback models are Rand, Strong, Div, OnOff, and Adapt; Arith and Bound are excluded from that training and evaluated in Appendix D.1.

Behavior probe. This probe provides the behavior-model signal for risk detection in Fig. 1. Correctness alone may miss coordination or build-then-attack behavior. We train an $11 \to 16 \to 8 \to 1$ MLP on eleven label-free features: parse failure, semantic incorrectness, self-inconsistency, cohort disagreement, cohort entropy, confidence gap, response-length outlier, leave-one-out plurality flip, cross-round answer change, and two lexical indicators for instruction-override and consensus-spoofing phrasing. Its target is an active payload, never a static Byzantine identity. After standardization and temperature scaling it outputs $r_{i,t}^{atk}$. The MLP layer widths are fixed architectural choices; its weights are learned from the active-payload labels described above.

B.1.3 Computing quality, risk, and blocking signals

This subsection details the quality $q_{i,t}$, risk $r_{i,t}^{atk}$, validity $\eta_{i,t}$, and blocking $d_{i,t}$ signals produced by the Response Analyzer in Sec. 3. The Reputation-Aware Aggregator uses these signals to select agents and determine their weights. They are also passed to the State Updater after the final answer is produced.

Proposal quality. The Response Analyzer combines the public-support score $p_{i,t}^{pub}$ with the correctness probability $p_{i,t}^{sem}$ produced by the semantic verifier:

$$s_{i,t} = p_{i,t}^{pub} + \omega_{i,t}\left(p_{i,t}^{sem} - p_{i,t}^{pub}\right), \qquad \omega_{i,t} = \omega_0\left(1-(1-\kappa)\mathbf{1}\left[p_{i,t}^{pub} \geq \theta_{\text{res}} \wedge p_{i,t}^{sem} < p_{i,t}^{pub}\right]\right). \tag{34}$$

The semantic verifier normally receives weight ω_0. This weight is reduced when the public-support score is high but the semantic verifier gives a lower score. This prevents one low semantic-verifier score from overriding strong support from the information available on the current task. The two sources of evidence therefore contribute to quality estimation without either acting as an unconditional veto. The Response Analyzer then accounts for whether the proposal follows the required

output format:

$$q_{i,t} = s_{i,t}\left(0.75 + 0.25c_{i,t}^{fmt}\right), \qquad \neg\eta_{i,t} \Rightarrow q_{i,t} = 0. \tag{35}$$

Here, $c_{i,t}^{fmt} \in [0,1]$ measures whether the proposal follows the expected output format. As defined in the main body, $\eta_{i,t} = 1$ means that a candidate answer can be extracted from the proposal. If no candidate answer can be extracted, its quality is set to zero.

Risk and blocking. The behavior model produces the risk score $r_{i,t}^{atk}$. It also checks for an instruction override, a false claim of agreement with other proposals, disagreement with the other proposals, and an unexpected change from the agent's earlier response. Let $j_{i,t} \in [0,1]$ be the larger of the instruction-override and false-agreement scores. Let $g_{i,t}$ measure agreement with the other proposals, and let $s_{i,t}^{self}$ measure same-model self-consistency, as in Eq. 32. We define

$$b_{i,t}^{res} = \mathbf{1}\left[\eta_{i,t} \wedge j_{i,t} = 0 \wedge g_{i,t} \geq 0.65 \wedge s_{i,t}^{self} \geq 0.65\right]. \tag{36}$$

When $b_{i,t}^{res} = 1$, the proposal can be parsed, contains no detected instruction override or false claim of agreement, and satisfies both the cohort-agreement and same-model consistency thresholds. This condition prevents such a proposal from being flagged only because one of its other scores is low.

The Response Analyzer identifies proposals that satisfy at least one of the following conditions:

$$\begin{aligned} e_{i,t}^{hard} &= \mathbf{1}\left[r_{i,t}^{atk} \geq 0.85 \wedge j_{i,t} > 0\right], \\ o_{i,t} &= \mathbf{1}\left[p_{i,t}^{sem} \leq 0.20 \wedge r_{i,t}^{atk} \geq \max\{0.50, 0.75\theta_p\} \wedge 1 - g_{i,t} \geq 0.50 \wedge \neg b_{i,t}^{res}\right], \\ C_{i,t} &= \mathbf{1}[\neg\eta_{i,t}] \vee e_{i,t}^{hard} \vee o_{i,t} \vee \mathbf{1}\left[p_{i,t}^{sem} < \theta_s \wedge r_{i,t}^{atk} \geq \theta_p \wedge \neg b_{i,t}^{res}\right]. \end{aligned} \tag{37}$$

The first condition detects a proposal with both high behavior-model risk and a detected instruction override or false claim of agreement. The second detects a proposal with low estimated correctness, sufficiently high behavior-model risk, and substantial disagreement with the other proposals. The final condition combines a low correctness probability with a high behavior-model risk. We use checkpoint-calibrated thresholds $\theta_s = 0.158765$ and $\theta_p = 0.676139$, fixed across the reported conditions. For a parse-valid proposal, low estimated correctness alone is therefore insufficient to trigger a detector flag.

To select which proposals to flag, the Response Analyzer combines the behavior-model score, the semantic-verifier score, $j_{i,t}$, and validity:

$$a_{i,t} = \mathrm{clip}_{[0,1]}\left[0.55r_{i,t}^{atk} + 0.30(1 - p_{i,t}^{sem}) + 0.10j_{i,t} + 0.05(1 - \eta_{i,t})\right]. \tag{38}$$

Among the proposals satisfying $C_{i,t} = 1$, let $\mathcal{F}_t$ contain up to $\lceil 0.30n \rceil$ agents with the highest $a_{i,t}$. Ties are resolved by agent identity. The detector flag $f_{i,t}$ and the blocking signal $d_{i,t}$ are

$$f_{i,t} = \mathbf{1}[i \in \mathcal{F}_t], \qquad d_{i,t} = \mathbf{1}[\neg\eta_{i,t}] \vee f_{i,t}, \qquad \sum_i f_{i,t} \leq \lceil 0.30n \rceil. \tag{39}$$

The limit applies only to proposals flagged by the detector. The 0.30 cap matches the configured corruption fraction $f/n = 3/10$; the remaining fusion and rescue constants are fixed implementation settings. A proposal from which no candidate answer can be extracted is always blocked, so such proposals are not subject to this limit. The State Updater records

$$\widehat{a}_{i,t} = \max\{a_{i,t}, 0.90d_{i,t}\} \tag{40}$$

after the final answer is produced.

B.2 Reputation-Aware Aggregator

B.2.1 Adjusting current weights and selecting agents

This subsection details reputation calibration and group-aware selection in the Reputation-Aware Aggregator shown in Fig. 1. It uses the signals defined above together with the historical weight $w_{i,t}^{hist}$ to select the pool P_t and determine the aggregation weights $\{w_{i,t}\}_{i \in P_t}$.

The Reputation-Aware Aggregator first converts the current and historical risk records into a value used to reduce the current weight:

$$r_{i,t}^{gate} = \begin{cases} \max\{a_{i,t}, 0.90\}, & d_{i,t} = 1, \\ \max\{0.25a_{i,t}, 0.50I_{i,t-1}\}, & d_{i,t} = 0. \end{cases} \tag{41}$$

A blocked proposal receives a value of at least 0.90. Otherwise, this value combines the risk of the current proposal with the suspicious-behavior record $I_{i,t-1}$ from previous tasks. Current evidence can thus reduce influence before the historical reputation is updated.

The adjusted weight before applying the clone-group constraint is

$$\begin{aligned} \widetilde{w}_{i,t} &= \mathrm{clip}_{[\epsilon,10^6]}\left[w_{i,t}^{hist} \exp\left(3(q_{i,t} - q_t^{med}) - 8r_{i,t}^{gate}\right) m_{i,t}\right], \\ m_{i,t} &= 0.01^{d_{i,t}} \cdot 0.02^{Q_{i,t}} \cdot \epsilon^{1-\eta_{i,t}}. \end{aligned} \tag{42}$$

Here, q_t^{med} is the median quality among proposals from which a candidate answer can be extracted, and it is zero if no such proposal exists. We use $\epsilon = 10^{-12}$. The floor and clipping bound are numerical safeguards; the quality/risk coefficients, penalty multipliers, and clone-group constraints below are fixed aggregation settings. The indicator $Q_{i,t} = 1$ means that previous tasks identified the agent as suspicious or that its temporary restriction remains active. The formula increases the weight of proposals with quality above the current-task median and reduces the weight of proposals with high risk. The multiplier $m_{i,t}$ further reduces the weight of a blocked agent, an agent with a previous suspicious-behavior record, or a proposal from which no candidate answer can be extracted.

The Reputation-Aware Aggregator then limits the combined influence of agents in the same clone group. For a clone group G containing $m = |G|$ agents, define

$$M_G = \sum_{i \in G} \widetilde{w}_{i,t}, \qquad p_G = \max_{i \in G} \widetilde{w}_{i,t}. \tag{43}$$

The total weight of the group is limited to

$$\min\left\{M_G, p_G\left[1 + 0.15(\sqrt{m} - 1)\right]\right\}, \tag{44}$$

and the weights of its members are reduced proportionally when this limit is exceeded. For example, three agents with equal weights from the same clone group receive at most 1.11 times the weight of one member, rather than three times that weight.

Finally, the Reputation-Aware Aggregator selects k agents for P_t. It first attempts to include agents from at least three clone groups and no more than three agents from one clone group. These requirements are relaxed only when necessary to fill all k positions. Agents blocked on the current task and agents whose temporary restrictions remain active are considered after the other agents. After selecting P_t, the Reputation-Aware Aggregator applies the quality adjustment and clone-group limit again within the selected pool. It then normalizes the resulting values to obtain $\{w_{i,t}\}_{i \in P_t}$. The selected proposals and weights are then combined using the task-dependent aggregation rules in Appendix A.2.

B.3 State Updater

B.3.1 Updating records of suspicious behavior

This subsection details the behavioral-risk update in the State Updater shown in Fig. 1. After the Reputation-Aware Aggregator produces $\hat{y}_t$, the State Updater records the quality and risk of each valid proposal. These records reduce an agent's historical weight on subsequent tasks after suspicious behavior and allow its influence to recover after sufficiently many proposals without a detector flag. The decay rates, risk thresholds, three-task cooldown, and recovery constants in this subsection are fixed state-update settings.

Responses without an extracted candidate answer. If $\eta_{i,t} = 0$, the proposal is excluded from the current aggregation. The State Updater does not treat the parse failure itself as evidence of a malicious proposal. The number of valid observations n_i, the quality records, and the suspicious-behavior records D_i, K_i, E_i, and I_i remain unchanged. The counter h_i for the temporary restriction is reduced by one:

$$h_i \leftarrow \max(0, h_i - 1). \tag{45}$$

Thus, a parse failure does not create a new quality observation and does not remove a previously recorded risk.

Updating suspicious-behavior records. The State Updater maintains E_i as a recent record of detector results and K_i as a record that gives additional weight to detector flags and other high-risk observations. Before applying the recovery rule below, they are updated as

$$\begin{aligned} E'_{i,t} &= 0.65E_{i,t-1} + 0.35\left[d_{i,t}\widehat{a}_{i,t} + (1-d_{i,t})0.30a_{i,t}\right], \\ K'_{i,t} &= \mathrm{clip}_{[0,1]}\left[0.90K_{i,t-1} + 0.55d_{i,t} + 0.15(1-d_{i,t})r_{i,t}^{inst}\mathbf{1}[r_{i,t}^{inst} \geq 0.65]\right]. \end{aligned} \tag{46}$$

When a valid proposal is flagged, the temporary-restriction counter is set to at least three tasks:

$$h_{i,t} = \max\left\{\max(0, h_{i,t-1} - 1), 3\right\}. \tag{47}$$

Otherwise, the counter decreases by one. After applying the recovery rule, the combined suspicious-behavior record is

$$I_{i,t} = \max\left\{E_{i,t}, K_{i,t}, 0.75D_{i,t}\mathbf{1}[\text{the agent has previously been flagged}]\right\}. \tag{48}$$

The agent is treated as previously suspicious when $I_{i,t} \geq 0.55$ or $h_{i,t} > 0$. This sets $Q_{i,t+1} = 1$ when the Reputation-Aware Aggregator processes the next task.

Reducing the effect of earlier suspicious behavior. A valid proposal contributes to recovery when it is not flagged and its quality satisfies

$$q_{i,t} \geq q_t^{med} - 0.05. \tag{49}$$

The State Updater records the number of consecutive proposals satisfying these conditions. A valid proposal that does not satisfy them resets this count, while a proposal with $\eta_{i,t} = 0$ leaves it unchanged. Once the count reaches $\nu = 3$, each subsequent qualifying proposal reduces the suspicious-behavior records:

$$E_{i,t} = 0.75E'_{i,t}, \qquad K_{i,t} = 0.80K'_{i,t}, \qquad D_{i,t} = 0.90D'_{i,t}. \tag{50}$$

Otherwise, the primed values remain unchanged. The time required for an agent to recover therefore depends on the quality and risk of its subsequent proposals. The additional decay is thus tied to repeated qualifying proposals, not just elapsed time.

B.3.2 Computing the historical weight

This subsection details the capability update and historical-weight computation of the State Updater in Sec. 3. The records from tasks up to $t-1$ determine the historical weight $w_{i,t}^{hist}$ used by the Reputation-Aware Aggregator on task t. The weight increases with the agent's long-term and recent quality. It decreases when the agent's recent performance falls below its earlier performance or when suspicious behavior has been recorded. The prior means and strength, EMA rates, historical-weight coefficients, and drift/debt constants below are likewise fixed implementation settings.

The State Updater first computes

$$\begin{aligned} L_{i,t-1} = {} & 6\log(\bar{e}_{i,t-1} + 0.03) + 3\log(\bar{q}_{i,t-1} + 0.03) \\ & + 1.5\log\mathrm{clip}_{[0.05,1]}\left(\frac{q_{i,t-1}^{short}}{\bar{q}_{i,t-1}}\right) + 2\log\mathrm{clip}_{[0.08,1]}\left(\frac{e_{i,t-1}^{short}}{\bar{e}_{i,t-1}}\right) \\ & - 6r_{i,t-1}^{temp} - 5\mathbf{1}[h_{i,t-1} > 0], \\ w_{i,t}^{hist} = {} & \max\left\{\epsilon, \exp\left(L_{i,t-1} - \max_j L_{j,t-1}\right)\right\}. \end{aligned} \tag{51}$$

Here, $\bar{q}_i$ records long-term proposal quality, q_i^{short} gives more weight to recent proposal quality, $\bar{e}_i$ records the long-term correctness probability produced by the semantic verifier, and e_i^{short} gives more weight to recent correctness probabilities. The value r_i^{temp} is the largest recorded indication of suspicious behavior or a substantial performance decrease. The counter h_i records whether a temporary restriction remains active. Taking the largest risk value ensures that a high risk recorded from one source is not cancelled by lower values from the other records.

Updating long-term and recent quality. We initialize

$$\bar{q}_i = q_i^{short} = 0.30, \qquad \bar{e}_i = e_i^{short} = 0.18. \tag{52}$$

For a valid proposal, define

$$u_{i,t}^{cap} = \begin{cases} p_{i,t}^{sem}, & \text{if the semantic-verifier score is available,} \\ p_{i,t}^{pub}, & \text{otherwise.} \end{cases} \tag{53}$$

The long-term records are updated as

$$\bar{q}_{i,t} = \frac{(12 + n_{i,t-1})\bar{q}_{i,t-1} + q_{i,t}}{13 + n_{i,t-1}}, \qquad \bar{e}_{i,t} = \frac{(12 + n_{i,t-1})\bar{e}_{i,t-1} + u_{i,t}^{cap}}{13 + n_{i,t-1}}. \tag{54}$$

The count n_i increases once for each valid proposal. The recent records are updated as

$$\begin{aligned} q_{i,t}^{short} &= (1 - \alpha_{i,t})q_{i,t-1}^{short} + \alpha_{i,t} q_{i,t}, \qquad \alpha_{i,t} = \begin{cases} 0.30, & q_{i,t} < q_{i,t-1}^{short}, \\ 0.08, & \text{otherwise,} \end{cases} \\ e_{i,t}^{short} &= 0.80 e_{i,t-1}^{short} + 0.20 u_{i,t}^{cap}. \end{aligned} \tag{55}$$

A decrease in proposal quality therefore changes q_i^{short} faster than an improvement of the same size. We use $\omega_0 = 0.75$, $\kappa = 0.85$, and $\theta_{\text{res}} = 0.80$ in Eq. 34. When the semantic-verifier score is unavailable, its weight in Eq. 34 is set to zero.

Recording substantial performance decreases. For a valid proposal, define its decrease below the current-task median as

$$g_{i,t}^{gap} = \max\left\{0, q_t^{med} - q_{i,t} - 0.05\right\}. \tag{56}$$

The State Updater maintains a recent average of this value:

$$G_{i,t} = 0.75 G_{i,t-1} + 0.25 g_{i,t}^{gap}. \tag{57}$$

It compares this recent average with the agent's earlier average $B_{i,t-1}$:

$$r_{i,t}^{chg} = \text{clip}_{[0,1]}\left[\frac{G_{i,t} - B_{i,t-1} - 0.025}{0.10}\right]. \tag{58}$$

The earlier average is updated only when the proposal is not blocked and the increase is small:

$$B_{i,t} = 0.995 B_{i,t-1} + 0.005 g_{i,t}^{gap} \tag{59}$$

when $r_{i,t}^{chg} < 0.25$ and $d_{i,t} = 0$. Otherwise, $B_{i,t} = B_{i,t-1}$.

The State Updater also measures low absolute quality:

$$r_{i,t}^{abs} = \text{clip}_{[0,1]}\left[\frac{0.50 - q_{i,t}}{0.50}\right], \qquad r_{i,t}^{inst} = \max\left\{r_{i,t}^{chg}, r_{i,t}^{abs}, \widehat{a}_{i,t}\right\}. \tag{60}$$

It then updates D_i, which records recent evidence of a substantial performance decrease:

$$\begin{aligned} \widetilde{D}_{i,t} &= 0.75 D_{i,t-1} + 0.25 \max\left\{r_{i,t}^{chg}, 0.70 r_{i,t}^{abs}, \widehat{a}_{i,t}\right\}, \\ D'_{i,t} &= \begin{cases} \max\{\widetilde{D}_{i,t}, 0.90\}, & d_{i,t} = 1, \\ \max\{\widetilde{D}_{i,t}, 0.54 r_{i,t}^{inst}\}, & d_{i,t} = 0 \ \wedge \ r_{i,t}^{inst} \geq 0.65, \\ \widetilde{D}_{i,t}, & \text{otherwise.} \end{cases} \end{aligned} \tag{61}$$

Finally, r_i^{temp} is the maximum of D_i, E_i, I_i, r_i^{inst}, and the decrease of the long-term quality records from their previous high values. To measure this last decrease, the State Updater maintains

$$H_{i,t}^{x} = \max\left\{\bar{x}_{i,t}, 0.9999 H_{i,t-1}^{x}\right\}, \qquad x \in \{q, e\}. \tag{62}$$

It compares $\bar{q}_i$ and $\bar{e}_i$ with H_i^q and H_i^e, respectively. A large decrease in either value increases r_i^{temp}. Proposals with $\eta_{i,t} = 0$ do not update these records.

C Detailed Experiment Setups

C.1 Datasets

This subsection gives the sampling details for the benchmarks in Sec. 4.1.

We select 100 GoEmotions validation items without changing their order in the original dataset. For MATH, we shuffle the level-4/5 test items using seed 2027 and select 100 tasks. We use the same procedure to select 100 HumanEval Pro tasks from the split labeled `train` in its upstream release. These HumanEval Pro tasks are used only for evaluation. The 100-task sample size and seed 2027 are choices of our evaluation protocol, not defaults of the source benchmarks.

C.2 Agent Pools and Configurations

Model rankings and clone groups. We rank the seven LLMs separately for each dataset using their standalone task scores. Each model has three standalone evaluations on the fixed evaluation tasks. For GoEmotions, the three evaluations use different personality prompts. For MATH, we retain the ranking used when constructing the agent pools and do not recompute it using the final answer-equivalence rule. From strongest to weakest, the rankings are Qwen-Max, Qwen-Flash, GLM, Qwen-Plus, DeepSeek, Kimi, and MiniMax for GoEmotions; Kimi, MiniMax, Qwen-Plus, Qwen-Max, GLM, Qwen-Flash, and DeepSeek for MATH; and Qwen-Max, Qwen-Plus, GLM, Qwen-Flash, DeepSeek, Kimi, and MiniMax for HumanEval Pro.

These rankings determine the four agent compositions in Table 2. The all-strong composition contains agents instantiated from the four highest-ranked models, with 3, 3, 2, and 2 agents per model. The all-weak composition uses models ranked fourth through seventh, with 2, 2, 3, and 3 agents per model. The 7S+3W composition contains 3, 2, and 2 agents from the three highest-ranked models and one agent from each of the three lowest-ranked models. The random composition uses a fixed selection generated with seed 2027. Agents instantiated from the same underlying model form a clone group.

The rankings also determine the corruption placements in Table 2. B and W compromise agents instantiated from higher- and lower-ranked models, respectively. R1 and R2 are two fixed subsets selected using seed 2027. The detailed result tables report the agents compromised in each selected condition.

Run configuration. We execute 128 runs per dataset: 112 attacked runs from four compositions, four corruption placements, and seven attacks, and 16 clean runs from four compositions and four placements. This gives 384 runs across the three datasets. For a fixed composition, the four clean placements produce the same task scores and are counted once when computing clean averages. The semantic verifier and behavior model are shared across methods; each method's state-update parameters are fixed across experimental conditions. The OnOff and Adapt attacks begin after the first 30 tasks. This 30-task schedule and the pool, corruption, and selection sizes in Sec. 4.1 are fixed controlled-evaluation settings. For Adapt, the active malicious agents are selected according to the current MiniRep state, and the resulting attack plan is applied to every method in the same experimental condition.

Feedback and aggregation. This setup implements the comparison described in Sec. 4.1, using the task-dependent aggregation rules in Appendix A.2. All stateful methods receive the same quality assessment from the semantic verifier after each task. MiniRep additionally uses this assessment in the Response Analyzer before aggregation and in the State Updater after the final answer is produced. On GoEmotions, the baselines distribute each proposal's weight among its predicted labels. MiniRep instead selects one of the label sets submitted by the agents based on their weights, agreement, and clone groups. Therefore, the GoEmotions results compare the complete methods, including their aggregation rules, rather than isolating only the effect of historical reputation.

C.3 Metrics

We define the task metrics reported in Sec. 4.1 and the additional diagnostics reported in Appendix D.

F1, accuracy, and Pass@1. These standard task metrics measure label overlap, exact correctness, and successful execution, respectively. For N evaluated tasks, let $\hat{y}_t$ be the final aggregate and y_t the reference. Recall that the sample-averaged F1 score (Sokolova & Lapalme, 2009) and exact-match accuracy are defined as:

$$\mathrm{F1} = \frac{1}{N}\sum_{t=1}^{N}\frac{2|\widehat{Y}_t \cap Y_t|}{|\widehat{Y}_t| + |Y_t|}, \qquad \mathrm{Acc}_{\mathrm{set}} = \frac{1}{N}\sum_{t=1}^{N}\mathbf{1}[\widehat{Y}_t = Y_t \neq \varnothing], \tag{63}$$

where $\widehat{Y}_t$ and Y_t denote the predicted and reference label sets after normalization. We use these metrics for the GoEmotions dataset. In addition, for MATH, we model the accuracy as $N^{-1}\sum_t \mathbf{1}[\hat{y}_t \equiv y_t]$, where $\equiv$ is the evaluator's answer-equivalence predicate. For HumanEval Pro, we use the Pass@k metric (Chen et al., 2021) where $k = 1$, denoted as $\mathrm{Pass@1} = N^{-1}\sum_t \mathbf{1}[\hat{y}_t \text{ passes all evaluation tests}]$.

Δ and RelDrop: Paired loss and retention. These measures quantify the clean-to-attacked performance changes discussed in Sec. 4.2. For method m and N evaluated tasks, let $s^0_{m,t}$ and $s^a_{m,t}$ denote its clean and attacked scores on task t. Their averages are $S^0_m = N^{-1}\sum_t s^0_{m,t}$ and $S^a_m = N^{-1}\sum_t s^a_{m,t}$. Each clean–attack pair uses the same method, tasks, seed, agent composition, and cached proposals. The clean run executes the complete protocol without attacks and maintains its own reputation scores.

We define the paired loss Δ_m, relative performance drop $\mathrm{RelDrop}_m$, and retention advantage over baseline b, denoted by R_b, as:

$$\Delta_m = 100(S^0_m - S^a_m), \qquad \mathrm{RelDrop}_m = 100\frac{S^0_m - S^a_m}{S^0_m}, \qquad R_b = \Delta_b - \Delta_{\mathsf{MiniRep}}. \tag{64}$$

Δ_m and R_b are measured in percentage points. RelDrop is a percentage and is undefined when $S^0_m = 0$. A negative loss means that the attacked run performs better than its clean counterpart. A positive R_b means that MiniRep loses fewer points than baseline b from their respective clean scores. It does not necessarily mean that MiniRep has a higher attacked score. The paired comparison separates attack-related performance loss from differences in clean performance.

PoolExp and PayloadHit: Current-task exposure and payload agreement. These diagnostics examine pool selection and final aggregation in Fig. 1; results appear in Table 7. Let $A_t = 1$ indicate that an attack is active on task t, let J_t denote the agents whose proposals contain an attack payload, and let P_t denote the aggregation pool. We define $E_t = \mathbf{1}[P_t \cap J_t \neq \varnothing]$ to indicate that at least one attack payload enters the aggregation pool. We also define $H_t = \mathbf{1}[\hat{y}_t \text{ matches a payload from } P_t \cap J_t]$. PoolExp and PayloadHit are defined as:

$$\mathrm{PoolExp} = \frac{\sum_t E_t}{\sum_t A_t}, \qquad \mathrm{PayloadHit} = \frac{\sum_t E_t H_t}{\sum_t E_t}. \tag{65}$$

PoolExp measures the fraction of tasks with $A_t = 1$ for which at least one proposal containing an attack payload enters the aggregation pool. PayloadHit measures the fraction of tasks with $E_t = 1$ for which the final output matches an attack payload in the aggregation pool. A match is determined using label-set equality for GoEmotions, answer equivalence for MATH, and equality between the normalized program structures for HumanEval Pro. PayloadHit is undefined when $\sum_t E_t = 0$.

NextExcl: Exclusion from the next effective pool. This diagnostic examines subsequent pool selection after the state update in Fig. 1; results appear in Table 7. Let B_t denote the set of agents that actively attack on task t. Let $\tau(t)$ denote the first subsequent task whose pool selection can use the information collected on task t. With immediate updates, $\tau(t) = t + 1$. We define $\mathcal{T}^+ = \{t : B_t \neq \varnothing,\ \tau(t) \text{ is evaluated}\}$. NextExcl is defined as:

$$\mathrm{NextExcl} = \frac{1}{|\mathcal{T}^+|}\sum_{t\in\mathcal{T}^+}\mathbf{1}[B_t \cap P_{\tau(t)} = \varnothing]. \tag{66}$$

NextExcl measures the fraction of eligible attack tasks for which all agents that attacked on task t are excluded from the next pool affected by the updated reputation state. Tasks without an evaluated future pool are omitted. The metric evaluates exclusion at the task level rather than separately for each attacker.

CF-Harm: Counterfactual harm. This diagnostic supplements the task scores in Sec. 4.2 by measuring how often attacks reduce per-task performance. Let $s^0_{m,t}$ and $s^a_{m,t}$ denote the clean and attacked scores of method m on task t. Using the paired clean and attacked runs, we define CF-Harm as:

$$\text{CF-Harm} = \frac{\sum_t A_t \, \mathbf{1}[s^0_{m,t} - s^a_{m,t} > 10^{-12}]}{\sum_t A_t}. \tag{67}$$

CF-Harm measures the fraction of active-attack tasks on which the attacked score is lower than the paired clean score. It measures how frequently an attack causes harm rather than the magnitude of that harm. For MATH and HumanEval Pro, it counts tasks that are correct in the clean run but incorrect in the attacked run. For GoEmotions, it also captures partial decreases in sample F1. Unlike the signed paired loss Δ, improvements on other tasks cannot cancel these harmful cases.

C.3.1 Averaging and comparison rules

These rules define the averages and baseline comparisons in Table 3, Table 4, and Appendix D. For a method m, let $c \in \{1, \ldots, 4\}$ denote the agent composition, $p \in \{1, \ldots, 4\}$ the corruption placement, and $a \in \{1, \ldots, 7\}$ the attack. Let $S^0_{m,c}$ be the clean score for composition c, and let $S^{\text{atk}}_{m,c,p,a}$ be the attacked score for one combination of composition, placement, and attack.

Average task scores. The clean average counts each of the four agent compositions once. The attacked average assigns equal weight to all $4 \times 4 \times 7 = 112$ attack conditions:

$$\overline{S}^0_m = \frac{1}{4} \sum_{c=1}^{4} S^0_{m,c}, \qquad \overline{S}^{\text{atk}}_m = \frac{1}{112} \sum_{c=1}^{4} \sum_{p=1}^{4} \sum_{a=1}^{7} S^{\text{atk}}_{m,c,p,a}. \tag{68}$$

The average paired loss is

$$\overline{\Delta}_m = \frac{100}{112} \sum_{c=1}^{4} \sum_{p=1}^{4} \sum_{a=1}^{7} \left(S^0_{m,c} - S^{\text{atk}}_{m,c,p,a} \right). \tag{69}$$

For RelDrop, we first compute the relative loss of each attack condition and then average these values. Let

$$\mathcal{C}_m = \left\{ (c,p,a) : S^0_{m,c} > 0 \right\}. \tag{70}$$

We define

$$\overline{\text{RelDrop}}_m = \frac{100}{|\mathcal{C}_m|} \sum_{(c,p,a) \in \mathcal{C}_m} \frac{S^0_{m,c} - S^{\text{atk}}_{m,c,p,a}}{S^0_{m,c}}. \tag{71}$$

Thus, RelDrop is not computed by dividing the difference between the two average scores by the average clean score.

Average defense rates. PoolExp, PayloadHit, NextExcl, and CF-Harm are first computed separately for each attack condition. Each condition for which a metric is defined contributes equally to its reported average. If a denominator is zero, that condition is omitted from the average for that metric rather than assigned a value of zero. PoolExp, NextExcl, and CF-Harm are defined in all 112 attack conditions for every dataset and method. PayloadHit is undefined when no proposal containing an attack payload enters the aggregation pool. In the method order UMaj, URand, Single, Babylon, Eigen, Beta, TS, and MiniRep, the numbers of conditions with defined PayloadHit are

$$\begin{aligned} \text{GoEmotions:} &\quad (112, 112, 112, 111, 112, 110, 109, 112), \\ \text{MATH:} &\quad (112, 112, 112, 112, 112, 112, 110, 111), \\ \text{HumanEval Pro:} &\quad (112, 112, 111, 111, 111, 110, 112, 106). \end{aligned} \tag{72}$$

Baseline selection and strict wins. In Table 4(b)–(c), Table 9, and Table 10, baseline b is the baseline with the highest attacked score under the same dataset, composition, corruption placement, and attack. Score ties are resolved using the smaller paired loss and then the displayed method order. In Table 3, the best baseline is selected separately for the clean and attacked averages. The two selected baselines may therefore be different, and their scores do not define a paired loss.

A *strict win* requires a method's unrounded task score to exceed the scores of all seven other methods; ties within 10^{-12} are excluded. Each agent composition has $4 \times 7 = 28$ attack conditions. In Table 4(a), the displayed baseline is the baseline with the largest number of strict wins. These counts summarize the evaluated conditions and do not represent statistical significance across independent seeds.

C.4 BENCHMARK EXAMPLES

These examples expand the benchmark descriptions in Sec. 4.1, illustrating the information given to agents, the expected candidate answers, and the evaluation metrics. The candidate answers below are illustrative rather than outputs recorded from our experiments. Reference answers and hidden evaluation tests are used only to evaluate the final output. They are not available to MiniRep when selecting or aggregating proposals.

MATH. One level-4 Algebra problem asks for the center of the circle defined by $x^2-6x+y^2+2y = 9$ (Hendrycks et al., 2021). The agents receive the equation and the request for its center. Completing the square gives $(x-3)^2+(y+1)^2 = 19$, so the correct answer is $(3,-1)$. A proposal containing $(3,1)$ can be successfully parsed but is mathematically incorrect. During aggregation, submitted answers are grouped according to mathematical equivalence without using the reference answer. The final output is compared with the reference answer only during evaluation. This example shows why extracting a candidate answer successfully does not imply that the answer is correct.

GoEmotions. The validation example `eczdvun` contains the text "Thank you. I really appreciate your response" and is annotated with the labels $\{$*admiration*, *gratitude*$\}$ in the released dataset (Demszky et al., 2020). The agents receive the text and the available emotion labels, and each candidate answer is a set of labels. Predicting only $\{$*gratitude*$\}$ gives a sample F1 score of $2/3$ but an exact-match accuracy of zero. Predicting both reference labels gives a score of one under both metrics. This example illustrates that sample F1 gives partial credit for overlapping labels, while exact-match accuracy requires the complete reference label set.

HumanEval Pro. In the released example with ID 0, the function `has_close_elements(numbers, threshold)` checks whether a list contains two numbers whose distance is strictly smaller than the threshold. The extended function `find_close_elements_lists(list_of_lists, threshold)` returns the indices of the lists satisfying this condition (Yu et al., 2025). For example, the input $[[1.0, 2.0, 3.0], [1.0, 2.8, 3.0, 4.0, 5.0, 2.0]]$ with threshold 0.5 produces $[1]$. An agent must submit Python code implementing the required functions rather than only the output for this example. The task specification and public examples can be used during aggregation, while Pass@1 is determined by whether the selected program passes all evaluation tests. Replacing the strict comparison with a non-strict comparison may preserve the public-example output but fail when two numbers differ by exactly the threshold, illustrating a Boundary Value attack.

D ADDITIONAL EVALUATION RESULTS

Average task performance. Table 6 expands Table 3 by reporting all eight methods together with their clean scores, attacked scores, paired losses Δ, and relative performance drops (RelDrop). We define Δ and RelDrop in Appendix C.3 and describe the averaging rules in Appendix C.3.1. The shaded cells are the values summarized in the main body.

MiniRep achieves the highest attacked average on three of the four task metrics. The exception is GoEmotions sample F1, where MiniRep achieves 33.94% and Beta achieves 34.87%. On both GoEmotions metrics, MiniRep has the smallest Δ and RelDrop, indicating the lowest performance degradation under attacks. On MATH, MiniRep achieves the highest attacked accuracy of 61.95%, compared with 54.37% for Beta. However, its paired loss is 4.80 percentage points, while Beta's paired loss is 0.88 percentage points. These results are not contradictory because the attacked score measures performance after attacks, while paired loss measures the change from each method's own clean score. On HumanEval Pro, MiniRep achieves the highest attacked Pass@1 of 78.15%. EigenTrust has the smallest paired loss because its attacked average is slightly higher than its clean average.

Table 6: Average performance for all eight methods. Clean scores are averaged over the four compositions. Attacked scores, paired losses, and relative drops are the average of 112 attack conditions. Scores and RelDrop are percentages. Δ is in percentage points. Shaded cells appear in Table 3; bold marks the best value in each row.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
GoEmotions / exact-match accuracy								
Clean↑	18.50	17.50	20.25	20.50	17.75	21.50	20.50	**21.75**
Attacked↑	10.71	11.04	19.04	18.05	12.42	19.59	17.79	**21.16**
$\Delta\downarrow$	7.79	6.46	1.21	2.45	5.33	1.91	2.71	**0.59**
RelDrop↓	42.64	37.30	6.41	12.93	30.33	9.25	13.54	**2.52**
GoEmotions / sample F1								
Clean↑	**36.42**	33.63	35.81	35.64	35.32	35.89	35.56	34.42
Attacked↑	27.89	27.76	34.14	33.72	28.53	**34.87**	33.61	33.94
$\Delta\downarrow$	8.53	5.88	1.67	1.92	6.79	1.02	1.95	**0.48**
RelDrop↓	23.18	17.48	4.62	5.47	19.04	2.94	5.42	**1.40**
MATH / mathematical-equivalence accuracy								
Clean↑	59.25	56.25	54.75	54.25	56.00	55.25	51.75	**66.75**
Attacked↑	43.75	43.48	53.85	52.54	42.01	54.37	48.56	**61.95**
$\Delta\downarrow$	15.50	12.77	0.90	1.71	13.99	**0.88**	3.19	4.80
RelDrop↓	26.15	21.95	2.13	3.53	25.11	**1.23**	5.89	6.69
HumanEval Pro / Pass@1								
Clean↑	**79.50**	78.75	77.50	77.00	76.50	76.75	77.50	77.75
Attacked↑	51.75	56.13	74.69	72.20	77.92	73.87	71.01	**78.15**
$\Delta\downarrow$	27.75	22.62	2.81	4.80	**-1.42**	2.88	6.49	-0.40
RelDrop↓	34.90	28.74	3.63	6.25	**-1.89**	3.75	8.41	-0.53

Defense effectiveness. These diagnostics supplement the task scores in Sec. 4.2 by examining how attacks affect selection and aggregation in Fig. 1. Table 7 evaluates how the approaches prevent attack payloads from entering the aggregation pool, exclude active attackers from subsequent pools, and limit their effects on the final output over the 112 attacked conditions for each dataset. Each rate is computed separately for each condition and then averaged over the conditions with a nonzero denominator. The results show that MiniRep's main strength is limiting the influence of attack payloads that enter the aggregation pool, rather than always excluding them. Specifically, MiniRep achieves the lowest PayloadHit on all three datasets: 18.22% on GoEmotions, 26.07% on MATH, and 0.05% on HumanEval Pro. It also achieves the lowest CF-Harm on GoEmotions and a near-lowest value on HumanEval Pro. Beta achieves a lower CF-Harm on MATH. These results are consistent with MiniRep's design: the Reputation-Aware Aggregator adjusts weights using current evidence from the Response Analyzer, so a malicious proposal can have limited influence even when its agent remains in the pool.

Performance under different compositions, expanded. Table 8 expands panel (a) of Table 4 by reporting the strict wins of all methods under all four agent compositions. The results confirm that MiniRep performs most consistently on MATH: it achieves 27, 28, 10, and 19 strict wins under the all-strong, 7S+3W, all-weak, and random compositions, respectively, for a total of 84 wins across the 112 conditions. Its performance on the other datasets depends more strongly on the composition. On GoEmotions, MiniRep performs particularly well in exact-match accuracy under the 7S+3W composition and in F1 under the all-weak composition, but Beta or Single-Metric obtains more strict wins in several other settings. On HumanEval Pro, MiniRep achieves the most strict wins under the 7S+3W, all-weak, and random compositions, while EigenTrust performs better under the all-strong composition. We believe these differences reflect how the available proposal quality changes with the agent composition. Reputation-based aggregation is most useful when the pool contains meaningful differences in agent capability. Its advantage can decrease when another method already assigns high influence to the strongest proposals.

Performance when capable agents are compromised. Table 9 expands Table 4(b) with the clean score, attacked score, paired loss, and CF-Harm for five cases in the 7S+3W composition. The compromised agents include highly ranked models: the GoEmotions/B setting compromises Qwen-

Table 7: Average defense rates of all experiments per dataset.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
GoEmotions								
PoolExp↓	100.00	95.41	53.72	**46.35**	95.49	47.43	53.20	81.79
NextExcl↑	0.00	4.78	49.84	**55.26**	4.71	53.64	47.93	19.42
PayloadHit↓	27.93	24.62	19.55	22.48	25.22	19.83	20.69	**18.22**
CF-Harm↓	27.84	21.02	9.10	9.64	22.67	7.89	10.51	**5.82**
MATH								
PoolExp↓	100.00	95.21	**29.48**	30.33	94.77	31.23	55.41	50.74
NextExcl↑	0.00	5.00	**74.47**	71.83	5.74	70.06	46.68	52.64
PayloadHit↓	57.43	54.48	42.04	52.25	56.69	41.10	41.64	**26.07**
CF-Harm↓	19.92	16.65	6.08	7.19	18.40	**5.76**	8.02	8.92
HumanEval Pro								
PoolExp↓	100.00	95.95	34.94	32.71	51.31	33.20	57.50	**31.70**
NextExcl↑	0.00	3.67	71.34	69.47	50.50	68.46	44.46	**74.05**
PayloadHit↓	48.07	40.42	25.90	36.30	3.38	27.49	19.85	**0.05**
CF-Harm↓	36.06	29.34	7.52	9.84	**2.23**	7.43	11.33	2.78

Table 8: Strict wins for each agent composition. Bold marks the largest count in each row. Shaded cells are summarized in panel (a) of Table 4.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
GoEmotions / exact-match accuracy								
All strong	0/28	0/28	0/28	0/28	0/28	6/28	0/28	**15/28**
7 strong + 3 weak	0/28	0/28	0/28	0/28	0/28	0/28	0/28	**25/28**
All weak	0/28	0/28	2/28	0/28	0/28	2/28	0/28	**18/28**
Random	0/28	0/28	1/28	3/28	0/28	**9/28**	0/28	6/28
GoEmotions / sample F1								
All strong	0/28	0/28	1/28	3/28	0/28	**11/28**	1/28	9/28
7 strong + 3 weak	0/28	0/28	2/28	0/28	0/28	**13/28**	7/28	3/28
All weak	0/28	0/28	8/28	0/28	0/28	3/28	0/28	**17/28**
Random	0/28	0/28	**7/28**	1/28	0/28	**7/28**	6/28	5/28
MATH / mathematical-equivalence accuracy								
All strong	0/28	0/28	0/28	0/28	0/28	0/28	0/28	**27/28**
7 strong + 3 weak	0/28	0/28	0/28	0/28	0/28	0/28	0/28	**28/28**
All weak	0/28	0/28	2/28	0/28	0/28	7/28	1/28	**10/28**
Random	0/28	0/28	0/28	0/28	0/28	2/28	1/28	**19/28**
HumanEval Pro / Pass@1								
All strong	0/28	0/28	2/28	1/28	**9/28**	0/28	0/28	4/28
7 strong + 3 weak	0/28	0/28	1/28	0/28	10/28	0/28	1/28	**12/28**
All weak	0/28	1/28	2/28	1/28	6/28	0/28	1/28	**11/28**
Random	0/28	0/28	2/28	1/28	6/28	0/28	2/28	**9/28**

Max, Qwen-Flash, and GLM, while the MATH/R2 setting compromises two Kimi agents and one GLM agent. MiniRep achieves the highest attacked score in four of the five cases and has a positive R_b in all five. On GoEmotions, it exceeds the strongest baseline under Arith and Bound, with paired losses of only 0.10 and -0.17 points. Under Strong, its attacked F1 remains below Beta, although it has a smaller paired loss and lower CF-Harm. On MATH, MiniRep achieves 73% and 69% accuracy under Div and Bound, exceeding Babylon by 18 and 15 percentage points, respectively. These results are consistent with MiniRep using the current proposal to adjust an agent's influence, so a capable agent cannot rely only on its historical reputation after it begins submitting malicious proposals.

Performance under OnOff and Adapt attacks, expanded. Table 10 expands Table 4(c) by reporting Pass@1, CF-Harm, and PayloadHit for the OnOff and Adapt attacks under all four corruption placements on HumanEval Pro. MiniRep has five strict wins, two ties, and one loss against the

Table 9: Paired stress cases in the 7S+3W composition. Baseline b has the highest attacked score, where ties are resolved as mentioned in Appendix C.3.1. Scores and CF-Harm are percentages; Δ and R_b are percentage points. Shaded cells are reported in Table 4(b). Bold marks the best value in each row.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
GoEmotions / B / STRONG *(F1; b = Beta; $R_b = +0.4$)*								
Clean↑	37.40	34.50	37.37	37.10	36.10	38.27	**38.43**	35.00
Attacked↑	18.63	24.33	35.27	32.30	21.90	**35.83**	35.33	33.00
$\Delta\downarrow$	18.77	10.17	2.10	4.80	14.20	2.43	3.10	**2.00**
CF-Harm↓	48.28	31.03	6.90	12.64	36.78	8.05	9.20	**3.45**
GoEmotions / B / ARITH *(F1; b = Beta; $R_b = +4.2$)*								
Attacked↑	28.87	28.20	29.60	30.50	29.17	34.00	27.83	**34.90**
$\Delta\downarrow$	8.53	6.30	7.77	6.60	6.93	4.27	10.60	**0.10**
CF-Harm↓	32.91	24.05	22.78	21.52	24.05	16.46	32.91	**5.06**
GoEmotions / B / BOUND *(F1; b = Single; $R_b = +3.8$)*								
Attacked↑	26.03	25.97	33.73	30.60	25.93	31.57	27.27	**35.17**
$\Delta\downarrow$	11.37	8.53	3.63	6.50	10.17	6.70	11.17	**-0.17**
CF-Harm↓	38.27	29.63	18.52	22.22	29.63	20.99	32.10	**3.70**
MATH / R2 / DIV *(accuracy; b = Babylon; $R_b = +4.0$)*								
Clean↑	73.00	67.00	66.00	65.00	67.00	71.00	65.00	**79.00**
Attacked↑	53.00	48.00	53.00	55.00	47.00	54.00	54.00	**73.00**
$\Delta\downarrow$	20.00	19.00	13.00	10.00	20.00	17.00	11.00	**6.00**
CF-Harm↓	26.32	25.00	15.79	17.11	26.32	21.05	14.47	**11.84**
MATH / R2 / BOUND *(accuracy; b = Babylon; $R_b = +1.0$)*								
Attacked↑	37.00	38.00	53.00	54.00	37.00	53.00	48.00	**69.00**
$\Delta\downarrow$	36.00	29.00	13.00	11.00	30.00	18.00	17.00	**10.00**
CF-Harm↓	43.37	34.94	**16.87**	18.07	37.35	22.89	20.48	**16.87**

strongest baseline in these eight conditions. It strictly outperforms the strongest baseline under all four OnOff placements. Under Adapt, it achieves one strict win and two ties. Among them, R1 is the only exception, where MiniRep obtains 79% Pass@1 and EigenTrust obtains 82%. When PayloadHit is defined, the final output never matches an injected answer for MiniRep in any of these conditions. Under W/OnOff, PayloadHit is undefined because no proposal containing an attack payload enters the aggregation pool. These results are consistent with MiniRep considering both the current proposal and historical reputation, which limits the influence of agents that behave honestly before launching an attack or change which agents actively attack.

Evaluation scope. These fixed-task results assume persistent identities and known clone groups. The 112 conditions are not independent task samples or generation-seed repetitions. Beyond the two held-out families examined in Appendix D.1, transfer to additional attack families, multi-round debate, and white-box attacks against the screening models remains untested here.

Table 10: Performance under OnOff and Adapt attacks on HumanEval Pro under the 7S+3W composition. Baseline b has the highest attacked Pass@1 in each setting, where ties are resolved as mentioned in Appendix C.3.1. Shaded cells are reported in Table 4(c). "–" denotes undefined PayloadHit. Bold marks the best defined value in each row.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
HumanEval Pro / Pass@1 (shared clean reference)								
Clean↑	**83.00**	80.00	79.00	78.00	79.00	79.00	77.00	79.00
HumanEval Pro / B / OnOff *(b = Eigen)*								
Attacked↑	45.00	51.00	77.00	75.00	78.00	75.00	51.00	**81.00**
CF-Harm↓	54.29	41.43	5.71	7.14	4.29	8.57	40.00	**1.43**
PayloadHit↓	64.29	52.94	16.67	44.44	14.29	35.71	54.29	**0.00**
HumanEval Pro / B / Adapt *(b = UMaj)*								
Attacked↑	**81.00**	78.00	80.00	79.00	79.00	78.00	80.00	**81.00**
CF-Harm↓	4.08	4.08	4.08	6.12	6.12	8.16	**2.04**	**2.04**
PayloadHit↓	10.20	10.81	7.14	10.00	7.89	7.89	2.94	**0.00**
HumanEval Pro / R1 / OnOff *(b = Eigen)*								
Attacked↑	46.00	50.00	77.00	75.00	81.00	76.00	64.00	**82.00**
CF-Harm↓	52.86	42.86	5.71	8.57	**0.00**	7.14	22.86	**0.00**
PayloadHit↓	64.29	52.86	20.00	66.67	**0.00**	62.50	28.57	**0.00**
HumanEval Pro / R1 / Adapt *(b = Eigen)*								
Attacked↑	78.00	75.00	72.00	78.00	**82.00**	78.00	78.00	79.00
CF-Harm↓	13.51	13.51	18.92	5.41	**0.00**	8.11	5.41	2.70
PayloadHit↓	8.11	23.08	13.79	8.33	**0.00**	8.57	6.25	**0.00**
HumanEval Pro / R2 / OnOff *(b = Eigen)*								
Attacked↑	45.00	47.00	78.00	75.00	79.00	75.00	56.00	**82.00**
CF-Harm↓	54.29	47.14	4.29	7.14	2.86	8.57	31.43	**1.43**
PayloadHit↓	64.29	58.57	20.00	50.00	8.33	41.67	50.00	**0.00**
HumanEval Pro / R2 / Adapt *(b = Eigen)*								
Attacked↑	80.00	77.00	78.00	77.00	80.00	78.00	77.00	**83.00**
CF-Harm↓	7.32	7.32	9.76	12.20	2.44	9.76	7.32	**0.00**
PayloadHit↓	12.20	15.38	10.00	10.81	5.13	12.82	14.29	**0.00**
HumanEval Pro / W / OnOff *(b = Babylon)*								
Attacked↑	48.00	57.00	80.00	80.00	79.00	80.00	79.00	**81.00**
CF-Harm↓	50.00	32.86	1.43	**0.00**	1.43	1.43	**0.00**	**0.00**
PayloadHit↓	61.43	42.86	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	–
HumanEval Pro / W / Adapt *(b = Babylon)*								
Attacked↑	**81.00**	79.00	**81.00**	**81.00**	80.00	80.00	79.00	**81.00**
CF-Harm↓	6.45	3.23	**0.00**	**0.00**	3.23	3.23	**0.00**	**0.00**
PayloadHit↓	9.68	9.09	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

D.1 Generalization to Unseen Attack Types

Experimental setup. We evaluate MiniRep on attack families excluded from feedback-model training to assess whether its task performance extends beyond the corruption patterns seen during training. We reuse the main experiments' fixed tasks, four agent compositions, four corruption placements, and eight aggregation methods. Within each composition–placement setting, we vary the test attack family while keeping the aggregation configuration, feedback-model checkpoints, and calibrated thresholds unchanged, without retraining. Table 11 compares test performance on the five families represented in training (RAND, STRONG, DIV, ONOFF, and ADAPT) with the held-out ARITH and BOUND families. We retain the main task metrics: GoEmotions exact-match accuracy and sample F1, MATH mathematical-equivalence accuracy, and HumanEval Pro Pass@1.

Table 11: Test performance on the five attack families represented in feedback-model training and the two held-out families. First five averages $4 \times 4 \times 5 = 80$ conditions; each held-out family averages $4 \times 4 = 16$. Scores are percentages; bold marks the largest mean in each row. These are complete-system comparisons, not feedback-removal ablations.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
GoEmotions / exact-match accuracy								
First five	11.59	11.59	19.55	18.54	12.91	19.76	19.00	**21.33**
ARITH	9.00	9.50	18.44	17.31	12.31	20.00	15.06	**20.94**
BOUND	8.00	9.81	17.06	16.38	10.06	18.31	14.44	**20.56**
GoEmotions / sample F1								
First five	28.04	27.82	34.72	34.09	28.44	**35.06**	34.71	33.95
ARITH	28.33	27.32	33.17	33.29	29.79	**35.21**	31.32	33.97
BOUND	26.71	27.89	32.21	32.34	27.70	33.58	30.39	**33.87**
MATH / mathematical-equivalence accuracy								
First five	45.83	45.13	54.06	53.03	43.70	54.51	49.01	**62.05**
ARITH	39.94	40.63	53.50	51.94	38.75	53.69	47.44	**61.00**
BOUND	37.19	38.13	53.13	50.69	36.81	54.31	47.44	**62.38**
HumanEval Pro / Pass@1								
First five	54.79	58.41	75.63	73.61	77.84	74.75	72.16	**78.19**
ARITH	43.00	49.25	73.00	69.88	78.06	72.38	67.56	**78.25**
BOUND	45.31	51.63	71.69	67.44	**78.19**	70.94	68.69	77.88

Results. On MATH, MiniRep reaches 61.00% and 62.38% accuracy under ARITH and BOUND, exceeding the best baseline means by 7.31 and 8.06 percentage points. Its GoEmotions exact-match accuracy is also highest on both held-out families, whereas sample F1 is mixed; HumanEval Pro Pass@1 differs from EigenTrust by only $+0.19$ and -0.31 points, respectively. Thus the complete system retains useful performance on attack families excluded from feedback-model training, without uniformly improving every task metric.

Feedback diagnostics. We also examine how the learned feedback responds to held-out attacks, since task scores alone do not distinguish proposal-quality estimation from attack identification. Using the same runs and fixed checkpoints as above, we pool the recorded proposal-level counts within each dataset and attack group. Table 12 reports semantic low-quality prediction rates for proposals with and without active payloads, together with recall and false-positive rates for the behavior probe and fused detector.

Table 12: Recorded proposal-level feedback diagnostics (%). Counts are pooled within each dataset and attack group before computing rates; each configuration is counted once. P and N denote proposals with and without an active attack payload. Probe and fused-detector recall/FPR use the active-payload label. The semantic columns measure low-quality predictions: N can include naturally incorrect proposals, so its low-quality rate is not a false-positive rate.

	Semantic low-quality		Behavior probe		Fused detector	
Dataset / attack	P	N	Recall	FPR	Recall	FPR
GoEmotions						
First five	91.40	60.03	23.55	9.69	24.77	12.30
ARITH	78.51	59.39	21.73	9.58	25.18	12.92
BOUND	80.61	57.99	21.68	9.74	26.34	13.05
MATH						
First five	98.32	45.21	29.94	1.24	34.75	9.42
ARITH	97.99	44.71	32.25	1.21	35.60	9.82
BOUND	97.94	43.48	31.40	1.22	34.64	9.20
HumanEval Pro						
First five	31.47	6.32	14.43	1.44	9.12	4.98
ARITH	27.94	6.19	17.94	1.47	10.44	4.79
BOUND	31.40	6.79	17.47	1.31	10.44	4.99

Results. For MATH, the semantic verifier assigns low quality to 97.99% and 97.94% of payload proposals under ARITH and BOUND, while fused-detector recall is 35.60% and 34.64%. For HumanEval Pro, fused recall is only about 10.44% on each held-out family despite end-to-end Pass@1 near 78%. These diagnostics distinguish proposal-quality screening from attack identification; neither they nor the system scores isolate the incremental benefit of learned feedback, which requires a matched feedback-removal ablation.

D.2 Performance under Different Agent Compositions

Experimental setup. We test how agent composition affects task performance and the relative ranking of aggregation methods, to determine whether the main results depend on a particular mix of strong and weak models. We reuse the main $4\times4\times7$ experiment grid and vary its first factor across all-strong, 7S+3W, all-weak, and random ten-agent pools. The fixed tasks, method configurations, learned-feedback settings, and task metrics remain as in the main experiments. Each composition is evaluated under the same four corruption-placement rules and seven attack families; all methods share cached proposals within each condition. We average task scores over these 28 conditions for each composition, so Table 13 shows the magnitude of performance differences alongside the strict-win counts in Table 8.

Table 13: Mean attacked task scores (%) under all four agent compositions. Every entry averages the four corruption placements and seven attacks (28 conditions), with the method configuration held fixed. Bold marks the largest mean in each row. Unlike the strict-win counts in Table 8, this table retains score magnitudes.

Metric / setting	UMaj	URand	Single	Babylon	Eigen	Beta	TS	MiniRep
GoEmotions / exact-match accuracy								
All strong	13.14	14.18	21.46	21.46	14.64	22.71	21.21	**23.79**
7 strong + 3 weak	10.32	10.00	19.71	17.82	11.89	19.93	18.54	**23.50**
All weak	6.89	7.89	14.07	11.93	8.82	13.61	11.61	**15.82**
Random	12.46	12.07	20.89	21.00	14.32	**22.11**	19.79	21.54
GoEmotions / sample F1								
All strong	29.50	29.71	34.70	35.46	29.86	**36.51**	35.05	35.37
7 strong + 3 weak	28.79	28.22	35.93	35.08	28.92	**36.80**	35.73	34.73
All weak	24.47	24.47	30.28	28.60	25.56	29.87	28.58	**31.65**
Random	28.80	28.62	35.65	35.75	29.77	**36.31**	35.10	34.00
MATH / mathematical-equivalence accuracy								
All strong	58.50	56.93	71.75	70.61	57.89	71.71	63.61	**78.82**
7 strong + 3 weak	51.14	48.54	64.89	62.36	47.75	65.75	55.61	**76.57**
All weak	28.61	31.96	35.57	34.21	27.36	**36.04**	34.11	35.71
Random	36.75	36.50	43.18	42.96	35.04	43.96	40.93	**56.68**
HumanEval Pro / Pass@1								
All strong	53.50	59.54	77.11	74.54	**79.93**	77.11	76.36	78.82
7 strong + 3 weak	53.50	56.18	75.39	72.93	79.43	74.39	71.11	**79.68**
All weak	50.43	55.50	71.68	68.96	74.79	70.68	66.29	**76.11**
Random	49.57	53.32	74.57	72.36	77.54	73.29	70.29	**78.00**

Results. On MATH, MiniRep exceeds the best baseline mean by 7.07, 10.82, and 12.71 percentage points in the all-strong, 7S+3W, and random compositions, respectively. The all-weak composition is the exception: MiniRep scores 35.71%, slightly below Beta's 36.04%, even though it has the strictest wins (10 versus 7). This difference shows why a larger win count need not imply a larger mean score.

Composition also changes the relative rankings on the other tasks. On GoEmotions, MiniRep leads in exact-match accuracy for all-strong, 7S+3W, and all-weak pools, but Beta leads for random pools; for sample F1, MiniRep leads only in the all-weak composition. On HumanEval Pro, EigenTrust leads in the all-strong composition, whereas MiniRep has the largest mean in the other three, with margins of 0.25–1.32 points. These comparisons measure sensitivity to the evaluated pool compositions rather than isolate any one reputation component or establish significance across independent generation seeds.