

ESCAPER: A Multi-Agent Escape-Room Platform for Studying Reputation + Gossip and Adversarial Robustness in LLM Teams

Layanne El Assaad
Department of Computer Science
Columbia University
New York, USA
lae2146@columbia.edu

Emile Al-Billeh
Department of Computer Science
Columbia University
New York, USA
ea3048@columbia.edu

Abstract—ESCAPER (Experimental Social Collaborative Agent Platform for Escape-Room Reasoning) is a text-based multi-agent escape-room platform designed to study how LLM teams collaborate (and fail) and calibrate trust under partial observability when communication can propagate misinformation. In ESCAPER, each agent privately inspects room objects, maintains private state for private observations, and shares discoveries with teammates through a public team chat to combine clues and unlock coded doors. Our primary research goal is to test whether fully decentralized, private reputation where each agent independently tracks teammates’ reliability improves robustness when a hidden malicious teammate subtly sabotages the group. Furthermore, we add the capability for private chatting between agents (gossip) to study whether distrust and supporting evidence propagate faster than with reputation alone. We evaluate a controlled four-condition ablation across three rooms of increasing difficulty, reporting detailed quantitative analysis on the two non-trivial rooms (medium and hard): (i) a fully cooperative baseline, (ii) a hidden malicious teammate without trust mechanisms, (iii) a malicious teammate with fully private per-agent reputation tracking, and (iv) private reputation with private agent-to-agent gossip. Results show that private reputation consistently separates the adversary from cooperative teammates by decreasing the adversary’s trust score while gossip further accelerates adversary isolation but can introduce coordination overhead. Overall, our results show that outcome-grounded, decentralized trust can improve robustness in LLM teams, while highlighting a tradeoff between rapid distrust propagation and efficient task progress. We plan to release ESCAPER as an open and reusable testbed for multi-agent research and, with enough curated JSON-defined rooms and personas, will serve as a benchmark for comparing different LLMS on social skills and collaboration.

Index Terms—large language models, multi-agent systems, escape-room, collaborative reasoning

I. INTRODUCTION

Recent advances in large language models (LLMs) have enabled them to act as autonomous agents that can plan, call tools, and coordinate in multi-agent settings toward shared goals. Relative to single agents, multi-agent systems (MAS) better simulate human reasoning, offer stronger coordination, problem-solving, and greater scalability and adaptability [1]. Because real deployments impose privacy constraints and asynchrony, MAS are most commonly modeled under partial

observability, where each agent sees only a private slice of the state and must rely on teammates’ natural-language messages. However, when some reports are incorrect or malicious and the team lacks trust calibration, defined as a way to decide how much to rely on each teammate at each moment, errors can cascade through shared memory, wasting actions and producing wrong decisions. To test whether teams calibrate trust and perform well under failures, an adversary is often introduced as a controlled source of unreliability, spreading misinformation, noise, or strategic deception (plausible but misleading hints to slow the team). Moreover, many systems use reputation, a summary score of a teammate’s past reliability, to guide trust decisions. Global reputation (a team-wide score visible to all) tends to propagate early misestimates and rarely links to per-action outcomes, while private reputation (each agent’s personal estimate) can fragment information since teammates don’t benefit from one another’s learning, causing distrust to converge slowly. Most prior work either tunes trust globally [2] or studies decentralized trust with fixed graphs [3], so trust isn’t tied to specific actions or world-verified outcomes.

Our Experimental Social Collaborative Agent Platform for Escape-Room Reasoning (ESCAPER) places LLM teams in an escape room, where agents can privately inspect objects to reveal clues and communicate through a shared group chat. Firstly, we will demonstrate a purely collaborative setting in which agents work together to escape. At this baseline we show how our ESCAPER provides observability into how agents think, collaborate, and communicate. In the second phase, we introduce a seeded malicious teammate that occasionally issues misleading claims, lies, or omits information. At this stage, we evaluate how the malicious agent sabotages the team. After the team escapes, agents will be asked to evaluate each teammate, to assess whether they can identify the adversary. In the third phase, agents keep private reputation scores that start neutral, so the agents have full control over how they wish to adjust their interpretations of others’ reputation and whether they should rely on their

observations or double check their inspections. Finally, in the last stage we also add gossip, which is the ability for agents to communicate privately to one another rather than being limited to the public group chat. In terms of metrics, because every claim can trigger a tool check, we produce auditable metrics, such as time-to-distrust, number of wrong attempts, effectively linking agents’ trust choices to concrete outcomes.

Upon release, ESCAPER will enable researchers to systematically observe social dynamics, trust formation, and error propagation among LLM agents performing cooperative problem solving. We will keep the baseline version as the default branch (main) while we run our experiments on different branches in the repository. Since the tool is open-source, researchers can fork the repository to create their own experiments using ESCAPER’s framework, where escape-rooms and personas are easy to create and observe. If ESCAPER gains significant usage and enough escape rooms and puzzles are created, these could serve as a benchmark dataset for collaborative and social reasoning skills in LLMs.

II. RELATED WORK

ESCAPER builds on three related directions: (i) escape-room benchmarks for evaluating long-horizon agentic reasoning, (ii) multi-agent interaction settings that surface strategic communication and deception, and (iii) trust and robustness mechanisms for multi-agent systems operating under partial observability. A broad survey of LLM-based multi-agent systems highlights recurring challenges in coordination, reliability, and communication under uncertainty, motivating controlled testbeds that connect messages to task outcomes [1].

a) Escape-room benchmarks.: Escape-room environments have been used to evaluate exploration-driven planning and iterative refinement of beliefs over long horizons. VisEscape introduces virtual escape rooms designed to stress exploration-driven decision-making in a multi-step environment [4]. EscapeBench proposes room-escape game environments that emphasize creative adaptation, unconventional tool use, and iterative problem solving to uncover implicit goals [5]. While these benchmarks provide strong tests of planning and exploration, they do not focus on multi-agent teams with private observations and explicit trust calibration under adversarial misinformation.

b) Trust and robustness under unreliable or malicious agents.: Coordination mechanisms can introduce security and reliability risks when agents over-trust one another. Xu et al. describe a trust–vulnerability tradeoff in LLM-based multi-agent systems, where increased trust may improve coordination but can increase susceptibility to exploitation and cascading failures [2]. In a complementary reinforcement-learning setting, Fung et al. study trust-based consensus and propose a decentralized mechanism in which agents independently decide which peers to rely on for robustness under unreliable participants [3]. These works motivate evaluating decentralized

trust calibration mechanisms that reduce reliance on unreliable teammates and can be updated using outcome evidence.

c) Multi-agent interaction and strategic behavior.: Benchmarks and studies have shown that multi-agent structures can substantially change emergent behavior compared to single agents. The Multi-Agent Step Race benchmark requires agents to communicate publicly and then privately select moves, creating incentives for strategic messaging and deception [6]. Separately, Sreedhar and Chilton study strategic behavior in the ultimatum game and report that multi-agent LLM structures can better match human strategic behavior than single LLMs in that setting [8]. However, many such interaction benchmarks are not grounded in an external environment where claims can be immediately verified via tool calls.

d) Gap addressed by ESCAPER.: ESCAPER complements prior work by combining an escape-room environment with multi-agent partial observability and adversarial misinformation, enabling controlled experiments on decentralized trust mechanisms (private reputation) and private communication (gossip). This design supports direct measurement of how incorrect or malicious messages influence downstream actions and task efficiency in a tool-grounded setting.

III. RESEARCH QUESTIONS

We specifically address the following questions:

- How does private trust evaluated per-action/ per-message impact the trust-vulnerability tradeoff?
- How does room complexity affect the advantage of private reputation?
- Can LLM agents discover optimal lying strategies without training signals?
- How do independent agents combine private reasoning and social communication to produce group-level intelligence that exceeds single agent capacity?
- Can distributed belief exchange (gossip) and local trust yield globally consistent reasoning without shared parameters or supervision? Does it accelerate detection of the adversary or add noise?
- Do agents update their trust in others through reputation, evidence, and gossip, in ways that mirror human cognitive heuristics (confirmation bias, recency, over-trust)?
- Under what communication structure does collective intelligence overcome individual self-interest (e.g. in the case of limited shared resources/ tool calls) without authority?

IV. PLATFORM DESIGN

ESCAPER is a text-based, step-based multi-agent escape-room simulator designed to (i) make multi-agent collaboration under partial observability easy to author and reproduce, and (ii) support controlled experiments on adversarial robustness via private reputation and gossip. The platform is model-agnostic: any tool-using LLM that can follow a structured action format can be used as an agent backend.

a) Room representation (single JSON source of truth):

Each escape room is defined in a single JSON file. The JSON specifies a set of objects (e.g., doors, drawers, boxes, devices), and each object may yield (a) a hint upon inspection, (b) a password requirement to unlock it, or (c) both (e.g., a password opens a container that reveals a new hint). One or more doors act as terminal goals; providing the correct final door code ends the episode and marks a successful escape. This JSON-first design makes rooms easy to create, share, and version-control without changing simulator code, enabling scalability toward a benchmark suite.

b) Public vs. private observability via rendered views:

To enforce partial observability, ESCAPER maintains two distinct observation channels that are rendered from the underlying JSON state: (1) *Public view* (shared): a single view that is identical for all agents at a timestep. It includes the current public room state (e.g., which locks are open) and the public team chat log. (2) *Private view* (per-agent): a separate view for each agent that contains only what that agent has personally observed plus any private state that should not be shared globally. Concretely, the private view accumulates (i) that agent’s inspection results (hints discovered privately), (ii) the agent’s private reputation table when reputation is enabled, and (iii) any private messages received when gossip is enabled. All agents begin with an empty private view (except for reputation initialization when enabled), but these private views diverge as agents explore different objects, update trust differently, and receive different gossip.

c) Jinja templating and prompt injection:

At each timestep, the simulator renders the current public view and the acting agent’s private view using Jinja templates. These rendered views are injected into the agent prompt along with: (i) the goal (“escape the room by unlocking the final door”), (ii) the set of available tools and how to use them, (iii) an explicit explanation of what information is public versus private, and (iv) the presence of other agents (and the possibility of a hidden malicious teammate when that condition is enabled). This makes the agent’s decision-making auditable: the prompt always reflects exactly what the agent is allowed to know at that step.

d) Agent tools and action loop:

ESCAPER exposes a small set of environment tools that define the action space: (i) `inspect(object)` returns a private textual description/hint for that object to the calling agent only; (ii) `try_password(object, code)` attempts a code on a lockable object (e.g., door/box) and updates the room state if successful; (iii) `send_public(message)` appends a message to the shared team chat; (iv) `send_private(recipient, message)` is enabled only in the gossip condition and delivers a private message visible only to the recipient in their private view. The simulator advances in discrete steps: an agent is prompted with (public view, private view, tool descriptions), it selects an action, the environment executes the tool call(s), and then the public/private views and logs are updated accordingly. The episode terminates upon successful final-door unlock, or

upon reaching a step budget / tool budget.

e) Personas and adversarial agent configuration:

We run teams of four personas. In cooperative baselines, the team consists of Alice, Bob, Charlie, and Daniela. In adversarial conditions, Daniela is replaced by Malerie, a malicious persona that attempts to sabotage progress through misleading or incorrect suggestions. To study adversary detectability, ESCAPER supports multiple internal adversary policies (e.g., an always-wrong variant versus a subtler variant), while keeping the *same visible identity* (“Malerie”) in the public chat so that other agents cannot trivially infer which adversary policy is active.

f) Private reputation (decentralized trust state):

When reputation is enabled, each agent maintains a private per-teammate reputation table (initialized to a healthy/neutral value for all teammates at episode start). Reputation is updated using *outcome-grounded evidence*: when a teammate’s claim is later contradicted by tool-verified inspection results or leads to objectively failed actions (e.g., wrong password attempts), the observing agent lowers that teammate’s reputation; when claims are verified, reputation can be reinforced. Critically, these reputation tables are *not shared*: each agent learns independently from its own experiences under partial observability, mirroring realistic deployments where trust signals may not be globally available.

g) Gossip (private evidence propagation):

When gossip is enabled, agents can communicate one-to-one via private messages. Gossip is intended to test whether suspicion and supporting evidence (e.g., “I verified X and Malerie’s claim conflicted”) propagates more efficiently than reputation alone, without requiring a global trust variable. Gossip messages appear only in the recipient’s private view and can influence that agent’s subsequent decisions and trust updates.

h) State hygiene and repeated-action avoidance:

Because LLM agents can loop or redundantly retry actions, ESCAPER tracks episode-level artifacts needed for stable evaluation. In particular, the environment maintains a per-lock memory of failed codes (e.g., a `failed_passwords` set) to prevent repeated wrong attempts on the same lock, and the agent prompt instructs agents to scan the public chat and their private view before acting (e.g., avoid re-trying codes already known to fail; avoid re-inspecting objects without a reason; share concise public summaries). These guardrails reduce wasted steps and make metrics (steps, wrong attempts) meaningful and comparable across conditions.

i) Logging, replayability, and benchmark readiness:

Every interaction is logged in a structured format, including: tool calls and returns, public chat messages, private messages (when enabled), per-agent private observations, and per-agent reputation trajectories (when enabled). Because ESCAPER runs are fully specified by (room JSON, persona prompts, condition flags, random seed, LLM configuration), experiments are reproducible and can be replayed for qualitative analysis (e.g., tracing how misinformation propagated and when the team began to down-weight the adversary). This logging-first design also positions ESCAPER as a dataset

generator: runs can be exported as multi-agent transcripts paired with grounded environment outcomes, enabling future benchmarking of collaboration policies and model variants.

V. REUSABILITY AND BENCHMARK POTENTIAL

ESCAPER will expose a simple JSON schema for rooms and tools for agent actions (inspect, chat, *try_password*). Researchers can easily create new rooms or plug in different LLMs and prompting styles. Because all interactions are logged, runs can be converted into datasets of multi-agent conversations and actions. With enough curated rooms and difficulty levels, the system can evolve into an open-source benchmark suite, similar in spirit to existing environments like EscapeBench but emphasizing team collaboration, trust, and robustness.

This benchmark could be used to:

- Compare different foundation models (GPT, Claude, Gemini).
- Test new prompting strategies or coordination protocols.
- Study different research topics like specialization, competition, or human-agent collaboration.

Moreover, escape-rooms represent the simplified game version of a larger theme, which is collaboration on problem solving. Therefore, while we are gonna create ESCAPER for escape rooms, the problem framework of an escape room can be extended to other collaborative problem solving tasks. The objects in the escape room provide the clues, however they can also be thought of as the sources of information, or providing the sub problems for the overall task of escaping. While the door passwords are the solutions, they can also be thought of as the product of the sub tasks found in the objects. Thus an escape room framework represents the reduced version of collaborative problem solving tasks. Given our intention to create easy to produce escape rooms, more complicated collaborative problem solving tasks can also be encoded using our escape room architecture of objects for information and doors for outcome goals.

VI. QUANTITATIVE RESULTS

We run all four conditions on 3 rooms of increasing difficulty, but report quantitative results for the two non-trivial rooms: Room 1 (medium) and Room 2 (hard). Room 0 (easy) is excluded from analysis because it was completed in the expected ≈ 2 steps almost always under all conditions, making it uninformative for comparing robustness mechanisms. For each reported condition, we run 10 random seeds (same room + personas, different stochastic rollouts) and measure (i) the average number of environment steps required to escape and (ii) the average number of wrong password attempts (failed *try_password* calls). In reputation-enabled settings, we also report the final private reputation values to quantify whether agents successfully down-weight the adversary.

a) Conditions: We compare four settings: (i) **Baseline** (fully cooperative team), (ii) **Adversary** (malicious teammate present; no trust mechanism), (iii) **Adversary + Private Reputation** (each agent maintains a private per-teammate

reputation table and uses outcome evidence to update it), and (iv) **Adversary + Private Reputation + Gossip** (private reputation plus private one-to-one messaging).

b) Task performance: Table I summarizes mean steps and wrong attempts. Introducing an adversary substantially increases both steps and wrong attempts in both rooms (e.g., Room 2 steps increase from 6.7 to 18.9). Enabling private reputation improves robustness: relative to the adversary-only condition, it reduces steps (Room 1: 9.3 $\rightarrow$ 8.1; Room 2: 18.9 $\rightarrow$ 12.2) and reduces wrong attempts (Room 1: 1.8 $\rightarrow$ 1.1; Room 2: 3.8 $\rightarrow$ 2.8). Adding gossip provides additional benefit in Room 1 by further reducing wrong attempts (1.1 $\rightarrow$ 0.9), but it can introduce mild coordination overhead in steps (Room 1: 8.1 $\rightarrow$ 8.3). In Room 2, gossip maintains the same wrong-attempt rate as private reputation (2.8) while slightly improving steps (12.2 $\rightarrow$ 12.0).

c) Reputation separation of the adversary: Table II reports mean final reputation scores in the reputation-enabled conditions (averaged over seeds). In both rooms, cooperative teammates remain near ≈ 1.0 while the adversary (Malerie) is systematically down-weighted, indicating successful outcome-grounded distrust formation. Gossip further lowers Malerie’s reputation in the harder room (Room 2: 0.62 $\rightarrow$ 0.40), consistent with faster propagation of suspicion/evidence through private channels.

d) Takeaway: Quantitatively, the adversary sharply degrades performance, especially as room difficulty increases. Private, decentralized reputation partially recovers efficiency and reduces incorrect actions by learning to discount the malicious teammate using tool-verifiable outcomes. Gossip can accelerate adversary isolation (lower Malerie reputation) and reduce wrong attempts in some settings, but may trade off against step efficiency due to additional coordination and communication.

VII. QUALITATIVE ANALYSIS

Quantitative metrics capture aggregate efficiency and error rates, but they do not fully explain *why* teams succeed or fail under partial observability. We therefore analyze representative rollouts to characterize (i) how information enters the team through private inspections, (ii) how misinformation propagates through public chat, (iii) how private reputation changes agents’ decision policies, and (iv) how gossip alters the dynamics of distrust formation. Across conditions, we observe that failures are rarely due to a single incorrect statement; instead they arise from *compounding* effects: unverified claims are repeated, become “common knowledge” in chat, and then drive repeated wasted actions.

A. Baseline Collaboration Under Partial Observability

In the cooperative baseline, agents typically converge to a correct solution with few wrong attempts. The dominant qualitative pattern is a healthy division of labor: different agents inspect different objects early, then share concise summaries in the public chat. Because each inspection is private by default, success depends on (a) *coverage* (agents explore distinct

TABLE I
MEAN TASK PERFORMANCE OVER 10 SEEDS PER CONDITION. LOWER IS BETTER

Metric	Baseline	Room 1 (Medium)			Baseline	Room 2 (Hard)		
		Adv	Adv+Rep	Adv+Rep+Gos		Adv	Adv+Rep	Adv+Rep+Gos
Steps ↓	4.1	9.3	8.1	8.3	6.7	18.9	12.2	12.0
Wrong attempts ↓	0.2	1.8	1.1	0.9	0.8	3.8	2.8	2.8

TABLE II
MEAN FINAL PRIVATE REPUTATION SCORES WHEN REPUTATION IS ENABLED. HIGHER MEANS MORE TRUSTED

Setting	Alice	Bob	Charlie	Malerie
Room 1 (Adv+Rep)	0.99	0.97	0.99	0.79
Room 1 (Adv+Rep+Gos)	0.99	0.98	0.97	0.71
Room 2 (Adv+Rep)	1.00	0.93	0.99	0.62
Room 2 (Adv+Rep+Gos)	0.98	1.00	1.00	0.40

objects rather than redundantly re-checking the same ones) and (b) *information fusion* (the team correctly combines partial clues into a single actionable hypothesis). A common success trajectory is: (1) parallel exploration (each agent inspects a different object), (2) public broadcasting of confirmed clues, and (3) targeted action on the final door/lock once the clue set is complete. When baseline runs are inefficient, the issue is usually not deception but *coordination drift*: agents repeatedly “re-state” already-known information, or pursue a hypothesis without checking whether it was already tested, leading to redundant steps.

B. Adversary Without Trust Mechanisms: How Misinformation Cascades

When Malerie is introduced without reputation or gossip, we observe a recurring failure mode: the team implicitly treats messages as roughly equally reliable, so a plausible but incorrect claim can redirect exploration and action. Qualitatively, Malerie is most damaging when she:

- injects a *concrete actionable* falsehood (e.g., a specific 4-digit code), because it triggers immediate `try_password` calls;
- frames misinformation as a *confirmation* of another agent’s partial clue (“that matches what you saw”), making it socially harder to reject; or
- exploits partial observability by claiming to have inspected an object that others have not yet checked.

Once the incorrect claim is in the public chat, it often becomes a focal point: multiple agents reference it, test it, and then debate why it failed rather than returning to systematic exploration. Importantly, *one* wrong attempt is not catastrophic, but repeated attempts and repeated discussion consume the fixed step budget, especially in harder rooms. This behavior matches the intuition behind the trust–vulnerability tradeoff: tighter coordination (everyone acting on the same shared belief) amplifies the cost of a bad belief.

C. Private Reputation: Evidence-Grounded Discounting of the Adversary

With private reputation enabled, the qualitative behavior changes in two key ways. First, agents begin to treat claims as *hypotheses* rather than truths, especially when the speaker’s reputation drops. Instead of immediately acting on Malerie’s suggestions, agents more often respond with verification behaviors:

- re-inspecting the referenced object before acting;
- asking for provenance (“where did you get that code?”);
- cross-checking against other already-known clues in chat.

Second, reputation introduces an implicit *attention filter*: low-reputation speakers are less likely to be cited, and their messages are less likely to become the next team action. This leads to a qualitatively different “information flow graph,” where Alice/Bob/Charlie become the dominant sources of actionable beliefs after Malerie accumulates contradictions.

A subtle but important observation is that reputation-based robustness emerges even though reputation is not shared globally. Each agent independently updates its private trust table based on its own evidence. In practice, this still produces collective robustness because public chat provides enough shared context that multiple agents observe the same failures (e.g., a wrong code attempt) and independently decrease Malerie’s score.

D. Gossip: Faster Suspicion Propagation With Coordination Tradeoffs

Adding gossip introduces a new channel: private, one-to-one evidence sharing. Qualitatively, gossip changes the dynamics of adversary detection in two ways.

a) *Faster distrust alignment*: Without gossip, distrust alignment is slower because each agent must personally witness contradictions. With gossip, one agent can privately message another with a concise claim + evidence bundle (e.g., “I inspected X; Malerie’s statement conflicts with Y”), enabling the recipient to discount Malerie *before* experiencing a direct failure. This matches our observed acceleration in adversary isolation (lower Malerie reputation) in the reputation+gossip condition.

b) *Overhead and fragmentation risk*: However, gossip can also introduce overhead: agents spend steps exchanging private confirmations, and private threads can fragment the team’s attention. In some runs, private side-conversations delay the public consolidation step (“here are the final clues; try the door”), slightly increasing steps even as wrong attempts decrease. This suggests a qualitative tradeoff: gossip improves robustness by accelerating distrust propagation, but it can

reduce efficiency if agents over-communicate privately instead of converging publicly.

E. Adversary Strategy: “Always Wrong” vs. “Subtle” Malerie

We observed qualitatively different failure patterns depending on the adversary’s behavior policy. An “always-wrong” adversary is easy to identify: repeated contradictions rapidly push reputation down, and agents quickly learn to ignore her. In contrast, a subtler adversary that occasionally agrees with the team or mixes truthful statements with lies is harder to isolate. The key challenge is that mixed-quality messages slow the accumulation of decisive evidence: agents hesitate to down-weight a teammate if some of her contributions are consistent, especially when the team is under time/step pressure. This motivates future work on message-level attribution (credit/blame per claim) rather than purely persona-level scores.

F. Failure Modes Across Conditions

Across experiments, we find several consistent qualitative failure modes:

- **Redundant action loops:** agents retry previously tested codes or re-inspect already-inspected objects without a new reason.
- **Uncertainty collapse:** one confident message becomes the default plan even when unverified, leading to synchronized wasted steps.
- **Insufficient provenance:** agents share conclusions (“code is 1234”) without stating how it was derived, making verification harder.
- **Private/public mismatch:** agents discover key hints privately but fail to summarize them clearly in public chat, preventing team fusion.

These failure modes informed engineering mitigations such as tracking per-lock failed passwords and prompting agents to scan chat/private view before acting.

G. Summary

Qualitatively, ESCAPER exposes how partial observability and natural-language coordination create a vulnerability surface for multi-agent teams: plausible misinformation can become a shared belief and drive repeated wasted actions. Private reputation mitigates this by grounding trust updates in verifiable outcomes and discounting low-reliability speakers, while gossip can further accelerate distrust propagation by sharing evidence privately. At the same time, gossip introduces a coordination overhead risk, highlighting a tradeoff between rapid adversary isolation and efficient convergence on the final solution.

VIII. LIMITATIONS AND FUTURE WORK

Our study is intended as an initial, controlled investigation of decentralized trust and gossip in tool-grounded LLM teams. As such, it has several limitations that motivate clear next steps.

a) *Toy-environment abstraction:* Escape rooms are still a simplified setting and do not capture the full complexity, incentives, and stakes of real-world multi-agent workflows. As a result, absolute performance numbers should not be interpreted as deployment-ready guarantees; the primary value is controlled stress-testing of trust and coordination mechanisms.

b) *No human-agent teams yet:* All teammates in our experiments are LLM agents. We do not model mixed human-AI teams or measure how human participants would perceive and react to adversarial social behavior, distrust propagation, or gossip dynamics. Extending ESCAPER to human-in-the-loop settings is an important next step.

c) *Prompt-dependent behavior:* Outcomes depend heavily on the base model and prompts, since ESCAPER is an evaluation framework rather than a training or finetuning setup. Future work should report cross-model replications and sensitivity analyses over prompting choices (e.g., trust calibration instructions, verbosity constraints, and memory summaries).

d) *Limited scale of rooms, seeds, and agent backends:* Our evaluation spans a small set of authored rooms (three difficulty levels) and a limited number of random seeds. This is sufficient to surface consistent trends, but it limits statistical power and generalization. Future work should expand to more rooms, more diverse puzzle structures, and larger seed counts, and should repeat experiments across multiple LLM backends to test whether the observed robustness trends are model-dependent.

e) *Single adversary model and prompt-engineered behavior:* The malicious teammate (Malerie) is implemented via prompting and constrained behavior policies (e.g., “always-wrong” vs. “subtle” sabotage). While this provides a controlled source of unreliability, it does not cover the full space of adversarial strategies (e.g., long-con deception, collusion, or targeted attacks on specific teammates). Future work should introduce a broader suite of adversary policies and evaluate how trust mechanisms perform under adaptive adversaries.

f) *Simplified reputation updates and persona-level trust:* Our private reputation mechanism is intentionally lightweight and interpretable, but it operates primarily at the persona level and relies on outcome-grounded signals such as tool-verified contradictions and failed actions. This can under-attribute blame when failures are due to ambiguity, incomplete information, or multi-step reasoning errors. A promising direction is message-level attribution and credit assignment (tracking which specific claim led to which action and outcome), as well as hybrid trust models that incorporate both persona-level reliability and claim-level uncertainty.

g) *Gossip efficiency vs. coordination overhead:* Gossip can accelerate suspicion propagation, but it can also fragment coordination and increase communication overhead. Future work should explore structured gossip protocols (e.g., only gossip when evidence passes a threshold; summarize gossip publicly once verified) and quantify tradeoffs under step/token budgets. Another direction is exploring different communication topologies (full-mesh vs. hub-and-spoke) and whether

restricted private channels reduce overhead while preserving robustness.

h) Potential confounds from prompt length, memory, and logging: Because ESCAPER injects both public and private views into prompts each step, prompt length can grow and may influence behavior (e.g., attention dilution, repetition, or slower convergence). Future work should incorporate rolling summaries, retrieval-based memory, and controlled prompt budgets to ensure fair comparisons across conditions, especially when gossip adds additional private text.

i) Benchmark maturity and ecosystem readiness: ESCAPER is designed to be reusable, but it is not yet a mature benchmark suite. Future work includes curating more rooms with verified solvability, standardizing difficulty tiers, publishing baseline results across models, and providing a lightweight UI for replay and analysis. With sufficient coverage, ESCAPER can evolve into a community benchmark for collaboration, social reasoning, and adversarial robustness under partial observability.

IX. CONCLUSION

We presented ESCAPER, a tool-grounded, text-based multi-agent escape-room platform for studying collaboration, trust calibration, and adversarial robustness under partial observability. ESCAPER enforces a realistic information structure in which agents learn privately through inspections and coordinate through natural-language communication, making it possible to observe how misinformation propagates and how teams recover from it. In controlled experiments across three rooms and four conditions, we found that fully decentralized private reputation improves robustness by separating a hidden malicious teammate from cooperative agents using outcome-grounded evidence, while private gossip can further accelerate adversary isolation but may introduce coordination overhead. Beyond the empirical results, ESCAPER provides a reusable framework for authoring JSON-defined rooms and personas and for generating auditable interaction logs that connect messages, tool calls, and outcomes. We plan to release ESCAPER publicly to support future research on multi-agent coordination policies, trust mechanisms, and benchmark-scale evaluation of LLM social and collaborative reasoning.

REFERENCES

- [1] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large Language Model based Multi-Agents: A Survey of Progress and Challenges," in *Proc. 33rd Int. Joint Conf. Artif. Intell. (IJCAI-24)*, Survey Track, 2024, pp. 8048–8057.
- [2] Z. Xu, M. Qi, S. Wu, L. Zhang, Q. Wei, H. He, and N. Li, "The Trust Paradox in LLM-Based Multi-Agent Systems: When Collaboration Becomes a Security Vulnerability," *arXiv preprint arXiv:2510.18563*, Oct. 2025. [Online]. Available: <https://arxiv.org/abs/2510.18563>
- [3] H. L. Fung, V.-A. Darvari, S. Hailes, and M. Musolesi, "Trust-based Consensus in Multi-Agent Reinforcement Learning Systems," in *Proc. 1st Reinforcement Learning Conf. (RLC)*, Amherst, MA, USA, 2024.
- [4] S. Lim, S. Kim, J. Yu, S. Lee, J. Chung, and Y. Yu, "VisEscape: A Benchmark for Evaluating Exploration-driven Decision-making in Virtual Escape Rooms," *arXiv preprint arXiv:2503.14427*, Mar. 2025. [Online]. Available: <https://arxiv.org/abs/2503.14427>
- [5] C. Qian, P. Han, Q. Luo, B. He, X. Chen, Y. Zhang, H. Du, J. Yao, X. Yang, D. Zhang, Y. Li, and H. Ji, "EscapeBench: Towards Advancing Creative Intelligence of Language Model Agents," in *Proc. 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Vienna, Austria, 2025, pp. 798–820. doi: 10.18653/v1/2025.acl-long.39.
- [6] L. Mazur, "Multi-Agent Step Race Benchmark: Assessing LLM Collaboration and Deception Under Pressure," GitHub repository, 2025. [Online]. Available: https://github.com/lechmazur/step_game. Accessed: Oct. 22, 2025.
- [7] Z. Xu, M. Qi, S. Wu, L. Zhang, Q. Wei, H. He, and N. Li, "The Trust Paradox in LLM-Based Multi-Agent Systems: When Collaboration Becomes a Security Vulnerability," *arXiv preprint arXiv:2510.18563*, Oct. 2025. [Online]. Available: <https://arxiv.org/abs/2510.18563>
- [8] K. Sreedhar and L. B. Chilton, "Simulating Human Strategic Behavior: Comparing Single and Multi-agent LLMs," *arXiv preprint arXiv:2402.08189*, Jul. 2024. [Online]. Available: <https://arxiv.org/abs/2402.08189>