International Journal of Mathematical, Engineering and Management Sciences
Vol. 10, No. 6, 1640-1657, 2025
https://doi.org/10.33889/IJMEMS.2025.10.6.078

Ram Arti Publishers

A Heterogeneous Multi-Agent Colluding Attack Defense System

Noah Oikarinen
Department of Electrical and Computer Engineering,
University of Massachusetts, Dartmouth, MA, USA.
E-mail: noikarinen@umassd.edu

Liudong Xing
Department of Electrical and Computer Engineering,
University of Massachusetts, Dartmouth, MA, USA.
Corresponding author: lxing@umassd.edu

(Received on June 6, 2025; Revised on July 18, 2025; Accepted on July 28, 2025)

Abstract
The voting-based redundancy mechanism is widely used as a fault-tolerant technique across various engineering and computing domains. However, this technique is vulnerable to colluding attacks where multiple malicious resources can collaborate to produce identical wrong results to potentially fail a task execution, posing significant threats to system integrity, performance, and reliability. This research introduces a heterogeneous multi-agent colluding attack defense system that employs a two-stage spot checking strategy within a credibility-based framework designed to detect and mitigate the impact of coordinated adversarial behavior within diverse agent environments. The proposed framework employs a dual-role architecture, consisting primarily of spotter and resource agents. Spotters are responsible for monitoring and evaluating the credibility of resource agents based on their performance and voting patterns. Resource agents that fail the first spot-check cannot participate in the voting process, while a second spot check targets a randomly selected agent from the majority vote, revoking the credibility of any colluding participants. The system dynamically and optimally allocates these agents under users' predefined cost constraints, balancing resource utilization and defense efficiency. By strategically adjusting the credibility scores of resource agents, the proposed defense mechanism adapts to ensure sustained system performance and reliability. Experimental studies are conducted to demonstrate effectiveness of the proposed heterogeneous multi-agent system in defending against colluding attacks in voting-based computing environments. The impacts of several key model parameters on system performance are also investigated.

Keywords- Colluding attack, Multi-agent system, Reliability, Spot-checking, Voting.

1. Introduction

Voting-based mechanisms (VBMs) are commonly used as fault-tolerant techniques in distributed computing systems, where each voting unit provides an output, and the collective votes determine the final result (Parhami, 1994). Popular strategies include majority voting where more than half the votes decide the outcome, plurality voting where the highest vote count determines the output and threshold voting requiring a predefined level of agreement to strengthen system integrity. VBMs have been successfully applied in numerous domains, such as imprecise data handling (Ivanov et al., 2016), safety monitoring and self-testing (Chaisawat and Vorakulpipat, 2020), multi-channel signal processing (Miao et al., 2024), pattern recognition (Tasci et al., 2021), target detection (Zhang and Zhou, 2023), image analysis (Iqbal et al., 2025; Pilar, 2025), node selection in Internet of Things (Chen et al., 2025), and blockchain consensus protocols (Gaur et al., 2025; Lei et al., 2025; Nguyen and Kim, 2018; Verma et al., 2025). More specifically, Chen and Avizienis (1978) introduced the idea of N-version programming, in which multiple independently designed software versions collectively vote to improve reliability and reduce single points of failure. Avizienis et al. (1971) examined a fault-tolerant computing design, the self-testing and repairing (STAR) computer, which used weighted voting to favor inputs from more reliable components. Similarly, Gogiashvili et al. (2000) addressed threshold-based redundancy in binary communication channels, and

Nordmann and Pham (1999) suggested a mathematical model to analyze reliability and cost of a weighted dynamic threshold voting system.

Although VBMs enhance system performance and fault tolerance, they remain vulnerable to collusion attacks. In these attacks, a subset of malicious voting units collaborates by sharing information or aligning their votes to undermine system integrity. When enough dishonest voting units collude, they can sway a vote toward incorrect results, amplifying the overall damage. Collusion attacks become more problematic when voting-based systems rely on the assumption that a majority or a threshold of votes are honest.

For effective analysis, a VBM-based system can be modeled as a multi-agent system (MAS) by treating each voting unit as an agent. An MAS architecture consists of multiple autonomous agents either homogeneous or heterogeneous, interacting within a shared environment. In general, MAS enable parallel task execution, enhance system resilience in the face of individual agent failures, and extend overall coverage across more complex tasks. MAS play a pivotal role in various applications, such as robotics (Burgard et al., 2005), grid computing (Wu et al., 2011), and wireless sensor networks (Tham and Renaud, 2005). Despite the benefits of MAS, integrating the VBM within an MAS poses unique challenges, particularly in coordinating agents to ensure accurate and trustworthy outcomes. The security and integrity of MAS are vulnerable to malicious agents that collude to manipulate outcomes, disrupt operations (Bonjour et al., 2022; Tassa et al., 2019) or obtain unauthorized access of the system. Having the ability to respond to colluding attacks in a heterogeneous multi-agent environment plays a significant role to ensure the reliability, integrity and effectiveness of these systems (Owoputi and Ray, 2022).

Previous research has explored a number of different strategies to detect and mitigate collusion in MAS. For example, Aguiar et al. (2022) introduced a data-driven contract design aimed at incentivizing agents to exert effort while detecting and disincentivizing collusion through dynamic contracts. Motwani et al. (2024) examined the potential for secret collusion among generative artificial intelligence (AI) agents, showing the need for a collusion mitigation countermeasure. Additionally, Bertrand et al. (2023) showed that Q-learning agents could learn to collude in iterated games, emphasizing the necessity for robust detection mechanisms. In the context of collusion prevention research for blockchain consensus protocol, Li et al. (2023) primary relied on smart contracts that forward information through a randomized communication topology, preventing malicious agents from exploiting predictable communication channels to coordinate collusion attacks. Wang et al. (2023) used an uncertainty and collusion proof Delegated Proof-of-Stake consensus mechanism centered around a selection pressure algorithm. This method allows voters to explore unfamiliar miners adaptively, thereby reducing wrong elections due to limited prior knowledge and uses a credibility system to penalize dishonest votes. Nevertheless, key challenges remain in designing dynamic, cost-aware frameworks capable of adaptively allocating resources under varying attack intensities. Additionally, there has been limited focus on heterogeneous agent roles specifically tailored to detect collusion and insufficient attention to credibility scoring with iterative reallocation, which is critical for sustaining a robust defense against collusive adversaries.

An effective method to detect dishonest or colluding agents in distributed environments is spot checking, in which a fraction of agents is randomly selected for verification to ensure they are honest agents or can produce correct outputs. Traditional approaches often adopt a single-stage spot check, verifying outputs once per task (Staab and Engel, 2009). This single-stage approach has been implemented for collusion tolerance in grid computing (Levitin et al., 2017, 2018), crowd sourcing (Wang et al., 2020), and peer-to-peer systems (Wang et al., 2018), where checks are performed to identify and isolate malicious or low credibility nodes. Particularly, Levitin et al. (2017, 2018) respectively modeled static and dynamic single-stage spot checking methodologies aimed at detecting and excluding colluding agents by determining the

optimal number of spotters and resource agents for the task. However, a limitation of single-stage spot checking is that colluding agents can still manipulate outcomes if they evade initial detection. Thus, a more robust defense mechanism is needed to address this vulnerability.

To fill the gap arising from the limitations of the single-stage spot checking, this paper contributes by proposing a two-stage spot checking approach in a heterogeneous multi-agent collusion attack defense system. The two-stage spot check introduces an additional verification step after the voting process, ensuring that even if colluding agents evade the initial spot check, they still face a second layer of scrutiny, thereby significantly reducing their chances of success. More specifically, during the first checking, each spotter verifies a single resource agent before the latter can participate in voting. Once a task vote is complete, the second spot check is performed on a randomly chosen agent from the majority voters. If this second spot check fails, the entire majority vote is deemed colluding and the credibility score of all agents in the majority is revoked. The proposed two-stage spot check strengthens collusion detection by verifying each prospective voting agent pre-task execution and providing a post voting verification step, thereby reducing the likelihood of undetected collusion influencing task outcomes. Additionally, the proposed defense system dynamically allocates the number of resource and spotter agents considering cost constraints and credibility scores, balancing resource usage and defense against collusion. In addition to the fundamental methodological novelty discussed above, the performance of the proposed framework in enhancing the resilience of VBM-based systems against collusion attacks is evaluated and validated using comprehensive experiments.

The rest of the paper is organized as follows. Section 2 describes the proposed collusion attack defense methodology. Section 3 evaluates performance of the proposed defense system under varying conditions such as agent reliability, agent costs, the number of colluding agents, credibility score adjustment combinations, total cost constraint, and initial credibility score. Section 4 concludes with the impact of the proposed work and future research that can be performed to extend the research.

2. Methodology

This section presents the theory behind the heterogeneous multi-agent colluding attack defense system. The methodology includes agent allocation, credibility score updating, collusion detection processes and the iterative operational framework that provides a robust defense against collusion attacks.

2.1 Defense Mechanism

The defense system operates within a multi-agent environment comprising a pool of heterogeneous agents, each initialized with a uniform credibility score (S_i). The credibility scores are updated based on task outcomes and detection of malicious behavior during the process.

The master agent begins by allocating the number of spotter (N_s) and the number of resource agents (N_r) under a specified cost constraint (C^*). This is mathematically represented by:

$$N_r^{max} = \left\lfloor \frac{C^* - N_s * C_s}{C_r} \right\rfloor \quad (1)$$

where, C_r and C_s are the cost associated with a resource agent and a spotter agent, respectively. Subsequently, N_r resource agents are selected from the pool of available agents with a positive credibility score. The probability (P_i) of any agent i being selected is proportional to its credibility score (S_i):

$$P_i = \frac{S_i}{\sum_{j=1}^{K} S_j} \quad (2)$$

where, K is the number of agents in the pool with $S_j > 0$. This probabilistic allocation ensures that agents deemed more credible are more likely to be assigned to participate in the task execution. Spotter agents have no credibility score because they are assumed to be perfect. Spotter agents are tasked with identifying colluding resource agents. During the first spotting phase, each spotter evaluates one resource agent by giving a task with a known correct answer. A resource agent becomes a work agent if its answer matches the desired correct answer; otherwise, the resource agent is identified as colluding and its credibility score is set to zero as in Equation (3), making it ineligible for future allocations by the master agent.

$$S_i^{new} = 0 \tag{3}$$

Task execution involves the participation of the selected resource agents with non-zero credibility scores. In other words, only the selected resource agents passing the first spot checking and those that are not selected for spot checking participate in task execution. These work agents provide votes on task outcome which are counted by the voting agent. A majority voting mechanism is employed to determine the task output when the number of working agents is at least 3. In the event of a tie, no output is determined and there is no credibility score update. If there are fewer than three work agents, the task still proceeds. When there are two work agents, they must both agree for the task output to be used. When only one work agent is available, its vote determines the task outcome.

To reduce the risk of the output being malicious (i.e., the majority of work agents participating in the task execution are colluding agents), a second spot checking phase is adopted. In this phase, an agent participating in the majority vote whose output matches the task output is randomly chosen for the spotter agent to check for colluding. If the selected agent is found to be a colluding agent, the task is deemed to be unsuccessful, and the credibility score of all winning agents in the majority vote are set to 0. This second spot checking phase serves as the second layer of defense for the system.

To determine the performance of the system, the task success metric (TSM) is defined as:

$$TSM = \begin{cases} \dfrac{Na}{N_v} & \text{if task succeeds} \\ -\dfrac{N_a}{N_v} & \text{if task fails} \end{cases} \tag{4}$$

where, N_a is the number of agents whose votes agree with the correct output and N_v is the total number of votes.

In the case of the task being deemed successful (i.e., passing the second spot checking), the credibility score of each agent participating in the majority voting is updated using Equation (5) or Equation (6). Specifically, when a work agent's output is correct (i.e., matching the voted output), its credibility score is incremented by a fixed value (ΔS) as in Equation (5):

$$S_i^{new} = S_i^{old} + \Delta S \tag{5}$$

Conversely, if a work agent's output is incorrect (i.e., not matching the voted output, which is the ground truth), its credibility score is decremented by a fixed value ($\Delta S'$) as in Equation (6):

$$S_i^{new} = S_i^{old} - \Delta S' \tag{6}$$

These updates ensure that agents contributing positively towards the task success are rewarded, while the colluding agents are systematically excluded from influencing future tasks.

The flowchart in **Figure 1** summarizes the defense process, which is listed as the following steps.

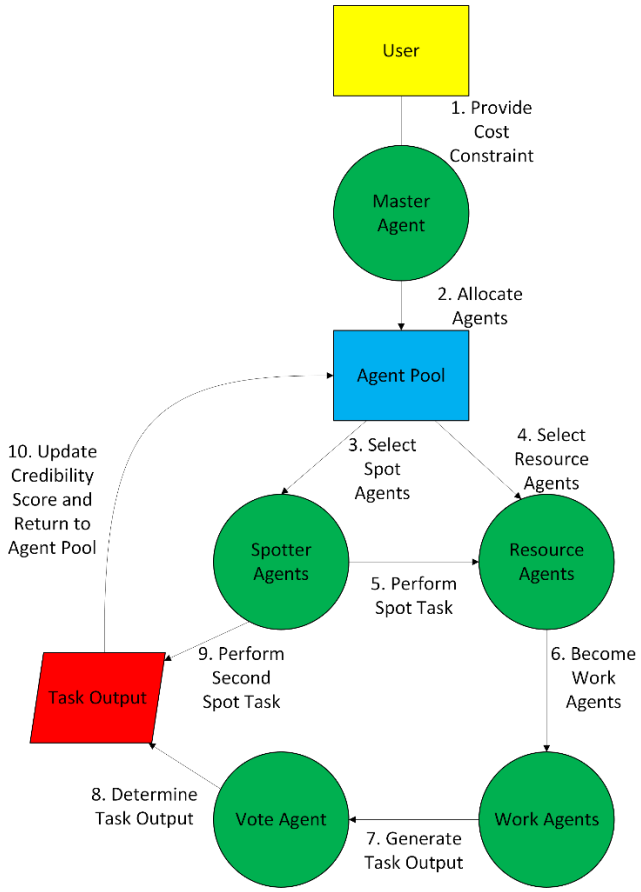

Figure 1. Flow chart of the proposed defense process.

1) The user provides the cost constraint (C^*) to the master agent.
2) The master agent determines the number of spotter agents (N_s) and resource agents (N_r) to allocate from the agent pool, constrained by the cost constraint (C^*).
3) The master agent selects (N_s) spotter agents from the agent pool.
4) The master agent selects (N_r) resource agents from the agent pool, each agent with a credibility score-dependent probability of being picked.
5) Each spotter agent performs the first spot checking task on a random resource agent.
6) Resource agents that are selected for the first spot check and pass the check are promoted to work agents. Resource agents that are not selected for the check are also promoted to work agents. Those that are selected but fail the check are excluded from participating in the task execution by having their credibility score set to zero.
7) Work agents participate in task execution and generate their task outputs.
8) A vote agent performs the majority voting of the task outputs from work agents and determines the final task output.
9) A randomly selected agent from the majority voting winning group is chosen for a second-stage spot check to validate the task output. If the selected agent fails the second spot check, all winning agents in

the majority voting group have their credibility score set to 0 and the task is considered failed. If the selected agent passes the check, the task is considered successful, and the credibility scores of all the winning work agents are increased by ΔS while the credibility scores of all the work agents whose output does not match the voted output are decreased by $\Delta S'$.

10) All participating agents are returned to the agent pool for potential reallocation in the next episode.

2.2 TSM Optimization Solution

The aim of the defense system is to maximize TSM subject to the total cost meeting the cost constraint, i.e., $C_{Total} \leq C^*$. In addition, the following constraints should be satisfied: $N_s \geq 1$, $N_r \geq 1$ and $N_r \geq N_s$.

To solve the TSM maximization problem, we implement the process in **Figure 1** iteratively over multiple episodes. Each episode begins with the user giving the cost constraint C^* to the master agent. As detailed in Section 2.1, following the agent's allocation, spot checking, task execution and voting, credibility scores are updated and colluding agents are possibly detected and isolated. The number of spotter agents is dynamically adjusted by the master agent in correlation to the task execution status. If the task is unsuccessful, the number of spotter agents to be allocated in the next episode is increased. Contrarily, if the task is successful, the number of spotter agents is decreased with a minimum number of spotter agents of one. This adaptive approach enables continuous refinement of the defense and maintaining a high success rate and robust detection capabilities across diverse and evolving multi-agent environments.

In addition to TSM, we measure the performance of the proposed defense system using reliability (R), as formulated as in Equation (7).

$$R = \frac{Total\ Number\ of\ Successful\ Episodes}{Total\ Number\ of\ Episodes} \tag{7}$$

3. Experiments and Evaluations

This section evaluates the performance of the proposed heterogeneous multi-agent colluding attack defense system and investigates the impact of several key parameters on the system performance, particularly on the system's ability to successfully detect and mitigate colluding agents. The task used for this experiment is a pattern recognition task where the agents vote on the classification for the task output. Two system performance metrics are evaluated, including TSM that quantifies the system's ability to correctly execute the task and R that provides an overall measure of the system's consistency in achieving successful outcomes across multiple episodes.

3.1 Parameter Setup

Each run involves 60 episodes. Each of the evaluation metrics is based off the average value over 1000 different runs. The default configuration is as follows:

- Honest Agent Reliability = 99.99%
- Initial Credibility Score = 5
- Cost of a Resource Agent (C_r) versus Cost of a Spotter Agent (C_s) = 2:1
- Number of Colluding Agents versus Number of Resource Agents = 5:15
- Credibility Score Updates $\Delta S = 3$, $\Delta S' = 2$
- Cost Constraint (C^*) = 10

Under the initial default configuration listed, the average TSM and R over 1000 runs can be obtained as 0.9870 and 0.9935, respectively.

In general, to achieve high TSM and R several parameters are found to be the most impactful (as detailed in the following subsections). High honest agent reliability is essential as it ensures that honest agents consistently produce correct outputs minimizing the influence of colluding agents. Assigning a lower initial credibility score allows the system to quickly identify and suppress colluding agents by limiting their early impact on task execution. Using harsher penalties (larger values of $\Delta S'$) for incorrect outputs could improve the system's ability to eliminate colluding agents and accelerate convergence. Regarding costs, better outcomes could be achieved when the master agent can afford enough spotters to effectively detect collusion while still maintaining a sufficient number of resource agents to accomplish tasks.

3.2 Impact of Honest Agent Reliability

The reliability of honest agents is varied from 5% to 99.99% in increments of 5%. This parameter gives the likelihood of honest agents providing the correct vote for the task execution. **Figure 2** and **Table 1** show that both TSM and R improve monotonically as honest agent reliability increases. When honest agent reliability is very low, below around 45%, TSM values are negative, indicating that colluding or unreliable agents frequently sway votes to incorrect results. As honest agent reliability crosses roughly 45%, TSM transitions into being positive, reflecting the system's growing ability to align the majority vote with the correct outcome. In parallel, R starts off low but also climbs progressively, surpassing 0.9 once honest agent reliability exceeds 80%. The trend highlights how even small increases in honest agent reliability can improve voting outcomes and increase system resilience against colluding attacks.

Table 1 also shows the results of TSM and R under both one-stage and two-stage spot checks. The two-stage procedure yields higher TSM and R in majority of cases. These increments imply that verifying a randomly chosen voter after the majority vote helps catch additional colluding agents, thereby raising TSM and R.

Table 1. TSM and R under different honest agent reliability.

Honest agent reliability	*TSM*		*R*	
	One-stage	Two-stage	One-stage	Two-stage
5.00%	-0.0663	-0.0667	0.0024	0.0025
10.00%	-0.0640	-0.0626	0.0059	0.0061
15.00%	-0.0569	-0.0577	0.0115	0.0105
20.00%	-0.0524	-0.0510	0.0172	0.0176
25.00%	-0.0465	-0.0471	0.0251	0.0239
30.00%	-0.0368	-0.0382	0.0349	0.0377
35.00%	-0.0280	-0.0266	0.0517	0.0535
40.00%	-0.0084	-0.0094	0.0849	0.0822
45.00%	0.0204	0.0220	0.1311	0.1340
50.00%	0.0204	0.0732	0.2268	0.2131
55.00%	0.1834	0.1944	0.3646	0.3827
60.00%	0.3257	0.3233	0.5438	0.5393
65.00%	0.4475	0.4564	0.6756	0.6856
70.00%	0.5689	0.5708	0.7926	0.7931
75.00%	0.6533	0.6553	0.8557	0.8592
80.00%	0.7263	0.7305	0.9028	0.9052
85.00%	0.7949	0.7970	0.9383	0.9407
90.00%	0.8596	0.8613	0.9668	0.9675
95.00%	0.9157	0.9181	0.9833	0.9843
99.99%	0.9675	0.9680	0.9898	0.9898

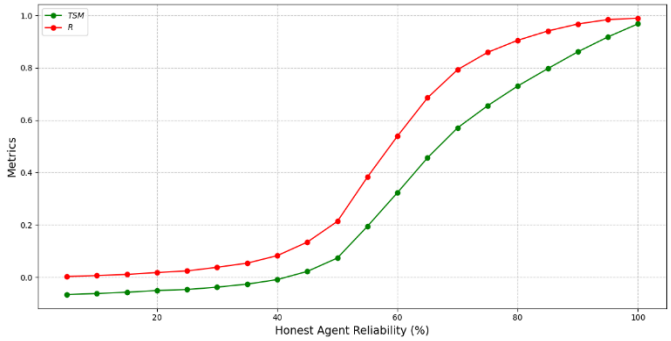

Figure 2. Impact of honest agent reliability.

3.3 Impact of Cost of Agents

Table 2 and **Figures 3** to **6** demonstrate how different $C_s : C_r$ combinations influence both TSM and R, also comparing single-stage and two-stage spot checks. Overall, the system maintains high performance at combinations with low agent costs, such as 1:1 or 2:1 or 1:2, where the master agent can afford enough spotters to effectively detect colluders while still maintaining sufficient resource agents to accomplish tasks and tolerate wrong outputs from any remaining colluders. Combinations with more expensive spotters like 3:1 experience a noticeable boost moving from one-stage spot check (TSM = 0.9284) to two-stage spot check (TSM = 0.9427) while balanced high spotter and resource agent costs like 3:3 remain the weakest performer in both TSM and R. Comparing one-stage to two-stage spot checks, improvements take place in both TSM and R for most cases. Overall, the results illustrate that the two-stage spot checking mechanism can refine the defense system's performance in general.

Table 2. TSM and R under different cost combinations.

$C_s : C_r$	TSM		R	
	One-stage	**Two-stage**	**One-stage**	**Two-stage**
1 : 1	0.9839	0.9840	0.9991	0.9991
2 : 1	0.9813	0.9814	0.9978	0.9979
1 : 2	0.9678	0.9679	0.9898	0.9900
3 : 1	0.9284	0.9427	0.9456	0.9603
2 : 2	0.9634	0.9650	0.9846	0.9862
1 : 3	0.9644	0.9653	0.9873	0.9874
3 : 2	0.8519	0.8589	0.8728	0.8800
2 : 3	0.9071	0.9150	0.9123	0.9201
3 : 3	0.3582	0.3591	0.3626	0.3635

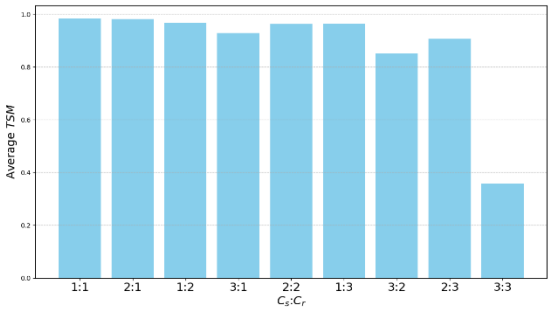

Figure 3. TSM under different cost combinations and single-stage spot check.

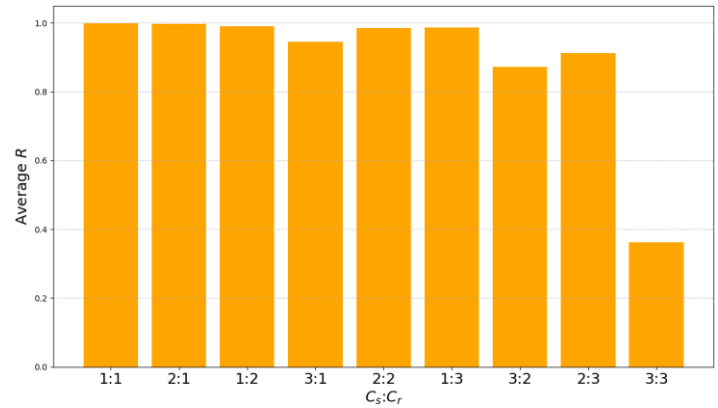

Figure 4. *R* under different cost combinations and single-stage spot check.

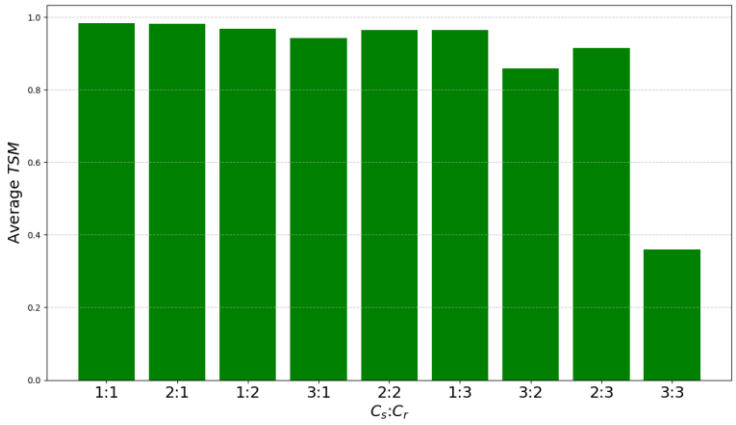

Figure 5. *TSM* under different cost combinations and dual-stage spot check.

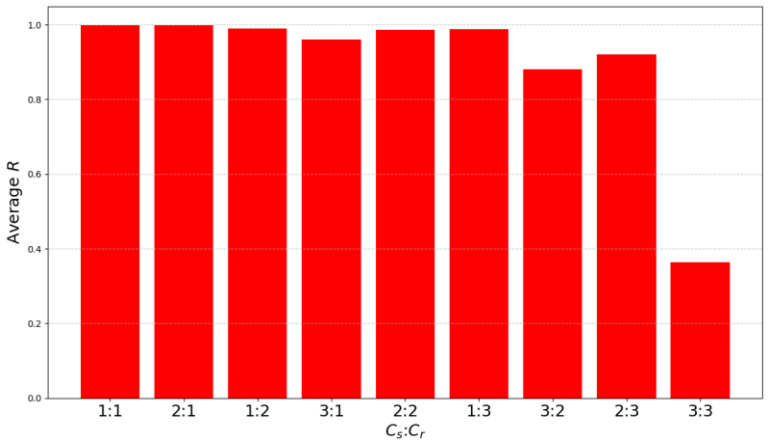

Figure 6. *R* under different cost combinations and dual-stage spot check.

3.4 Impact of Number of Collusion Agents

Assuming there are 15 resource agents in the pool. We evaluate the system's resilience against varying levels of collusion attacks by considering the number of colluding agents from 1 to 15. **Table 3** summarizes the values of TSM and R under both one-stage and two-stage spot checking mechanisms. **Figure 7** shows these results graphically, with a zoomed-in view presented in **Figure 8**.

Table 3. TSM and R under different numbers of colluding agents.

Number of colluding agents	TSM		R	
	One-stage	Two-stage	One-stage	Two-stage
1	0.9953	0.9955	1.0000	1.0000
2	0.9897	0.9898	0.9990	0.9991
3	0.9838	0.9842	0.9974	0.9978
4	0.9756	0.9769	0.9937	0.9945
5	0.9678	0.9691	0.9900	0.9904
6	0.9594	0.9603	0.9849	0.9851
7	0.9504	0.9539	0.9794	0.9815
8	0.9418	0.9472	0.9734	0.9765
9	0.9340	0.9399	0.9678	0.9703
10	0.9272	0.9354	0.9622	0.9660
11	0.9220	0.9330	0.9563	0.9626
12	0.9201	0.9315	0.9515	0.9583
13	0.9170	0.9287	0.9426	0.9508
14	0.8683	0.9177	0.8876	0.9360
15	-0.0167	-0.0167	0.0000	0.0000

It can be observed that, regardless of whether one-stage or two-stage spot checking is used, both TSM and R remain high when the number of colluding agents is relatively low compared to the number of honest agents. As the number of colluding agents increases, both metrics decrease. However, incorporating a second spot check provides a moderate improvement in TSM and R with especially pronounced gains at higher collusion levels. In particular, with 14 colluding agents, TSM improves from 0.8683 to 0.9177 and R from 0.8876 to 0.9360 when the second spot check is implemented. The reason TSM and R remain high even with a large number of colluding agents is due to the rapid elimination of these agents in early episodes. After the first few episodes, most or all colluding agents are identified and assigned a credibility score of zero through the spot checking processes, making them ineligible for future allocation. As a result, a single honest agent is sufficient to maintain correct task outcomes in later episodes. This leads to a high number of successful episodes, which gives high values of TSM and R. With 15 colluding agents, both one-stage and two-stage cases collapse to TSM = -0.0167 and R = 0, indicating a tipping point at which malicious agents completely dominate. Overall, the data shows that dual spot checks consistently yield higher TSM and R than a single spot check.

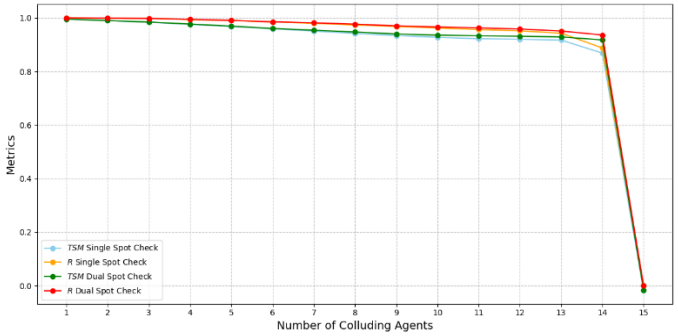

Figure 7. TSM and R under different numbers of colluding agents.

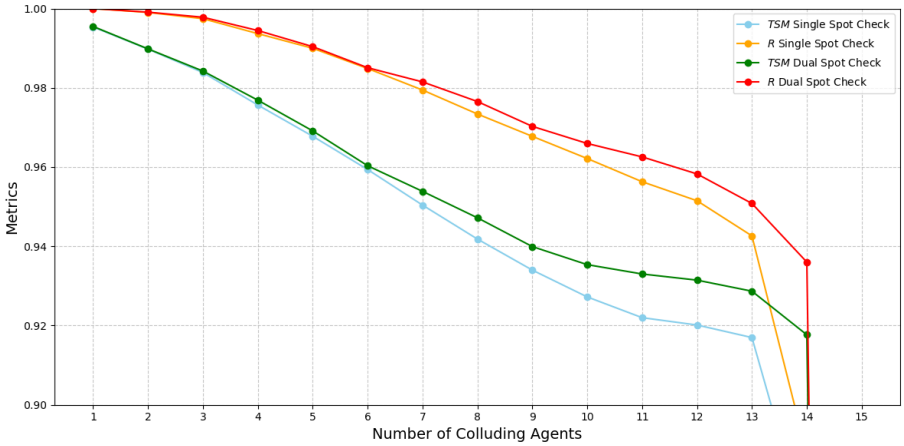

Figure 8. TSM and R under different numbers of colluding agents (zoomed-in view).

3.5 Impact of Credibility Score Adjustment

Table 4 summarizes the values of TSM and R under both one-stage and two-stage spot checking mechanisms, showing that both metrics remain high across all credibility score update combinations.

Table 4. TSM and R under different credibility score adjustment combinations.

$+\Delta S/-\Delta S'$	TSM		R	
	One-stage	Two-stage	One-stage	Two-stage
+1/-1	0.9500	0.9530	0.9841	0.9849
+1/-2	0.9588	0.9611	0.9864	0.9875
+1/-3	0.9658	0.9668	0.9889	0.9890
+1/-4	0.9684	0.9692	0.9903	0.9902
+1/-5	0.9786	0.9784	0.9922	0.9922
+2/-1	0.9544	0.9567	0.9863	0.9870
+2/-2	0.9626	0.9638	0.9882	0.9889
+2/-3	0.9678	0.9688	0.9898	0.9902
+2/-4	0.9710	0.9715	0.9913	0.9913
+2/-5	0.9790	0.9790	0.9930	0.9929
+3/-1	0.9595	0.9604	0.9888	0.9886
+3/-2	0.9659	0.9657	0.9905	0.9899
+3/-3	0.9701	0.9702	0.9910	0.9911
+3/-4	0.9735	0.9729	0.9924	0.9919
+3/-5	0.9790	0.9791	0.9927	0.9931
+4/-1	0.9613	0.9628	0.9899	0.9895
+4/-2	0.9669	0.9679	0.9905	0.9908
+4/-3	0.9710	0.9716	0.9915	0.9918
+4/-4	0.9740	0.9747	0.9923	0.9927
+4/-5	0.9798	0.9798	0.9937	0.9936
+5/-1	0.9640	0.9652	0.9899	0.9906
+5/-2	0.9687	0.9696	0.9910	0.9915
+5/-3	0.9722	0.9730	0.9920	0.9923
+5/-4	0.9757	0.9757	0.9932	0.9931
+5/-5	0.9799	0.9802	0.9936	0.9939

Figures 9 to **12** demonstrate how different credibility score adjustment combinations influence TSM and R in bar graphs, also comparing one-stage (**Figures 9** and **10**) and two-stage (**Figures 11** to **12**) spot checking. There are some fluctuations ranging from about 0.95 to 0.98 in TSM and around 0.98 to 0.99 for

R. Larger negative increments like -5 do appear to penalize colluding agents more effectively, boosting overall performance. The difference between one-stage and two-stage spot check is a gain of a fraction of a percentage point in both metrics. For instance, +1/-1 moves from $TSM = 0.9500$ to 0.9530 and $R = 0.9841$ to 0.9849. While under the +5/-5 combination, changes are minimal. These improvements do demonstrate that the second spot check can further tighten defense against colluding agents, especially when being coupled with harsh penalties for incorrect outputs.

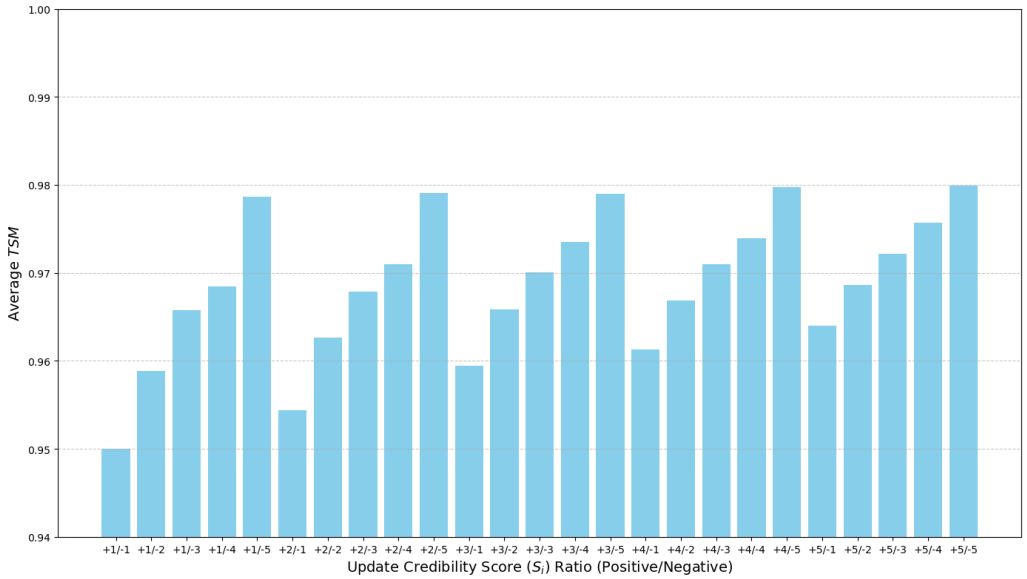

Figure 9. *TSM* under different credibility score adjustment combinations and single-stage spot check.

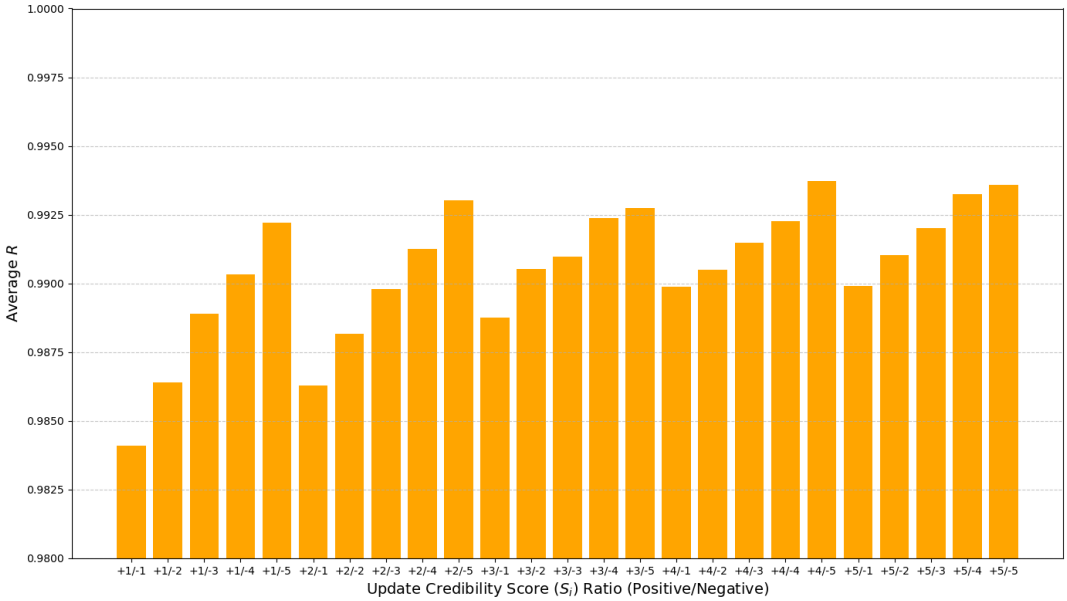

Figure 10. R under different credibility score adjustment combinations and single-stage spot check.

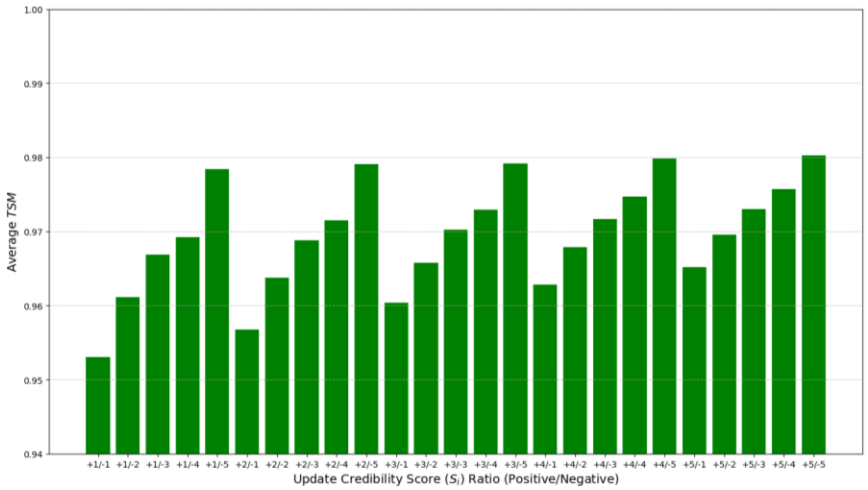

Figure 11. TSM under different credibility score adjustment combinations and dual-stage spot check.

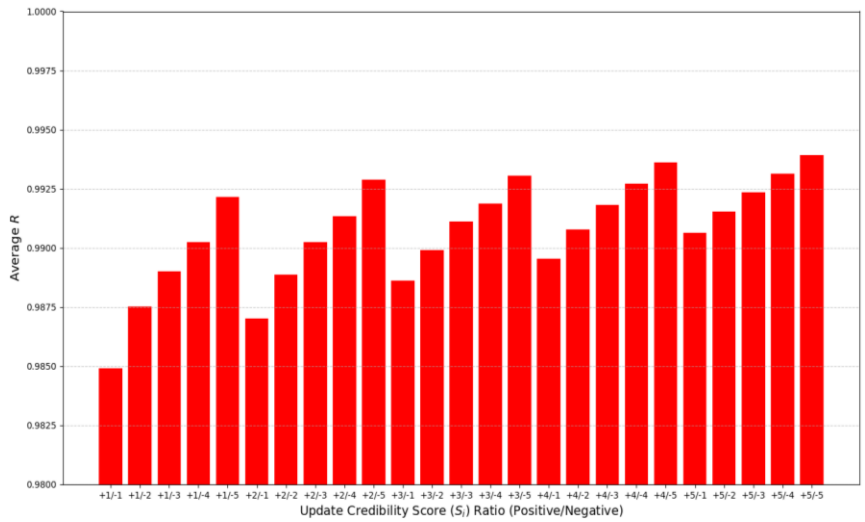

Figure 12. R under different credibility score adjustment combinations and dual-stage spot check.

3.6 Impact of Cost Constraint

The cost constraint is varied from 1 to 15 to evaluate the system's performance under different levels of budget restraints. This parameter dictates the total resource available for allocating spotter and resource agents in each episode.

Figure 13 shows the values of TSM and R under varying cost constraints for the dual spot checking mechanism. **Table 5** presents the values of TSM and R for both single and dual spot checking mechanisms. Under both mechanisms, the system fails entirely with $TSM = 0$ and $R = 0$ when the cost constraint C^* is 3 or below, as there is not enough resource to allocate the minimal combination of spotters and resource agents. At $C^* = 4$, both TSM and R jump to about 0.35, signaling that the master agent can now deploy enough agents to detect some collusion and achieve moderate task success. Above that budget constraint,

the metrics rise sharply by $C^* = 6$, at which TSM surpasses 0.93, and R climbs above 0.94. From $C^* = 7$ onward, both TSM and R stabilize near or above 0.96. Comparing the single and dual spot checks, there is not much of a difference between the two methods. After $C^* = 6$ with most of the time the single spot check preforms better than the dual spot check. These results show a lower bound on the total resource cost for effective spotter and resource agent allocation and show the near optimal performance of the system when C^* is above 6.

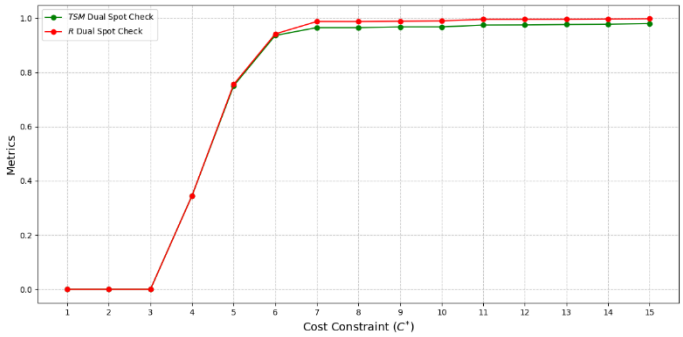

Figure 13. TSM and R under varying cost constraints.

3.7 Impact of Initial Credibility Score

The initial value of S_i is varied from 1 to 20 to evaluate the system's performance under different levels of initial trust of each agent.

Figure 14 and **Table 6** demonstrate that both TSM and R begin at relatively high levels for low initial S_i then steadily decrease as S_i increases. For instance, at initial $S_i = 1$, TSM is around 0.98 and R is around 0.995. For $S_i = 20$, TSM falls to about 0.945 and R drops towards to around 0.981. This pattern suggests that assigning higher initial S_i makes it slightly harder to distinguish honest agents from potential colluding agents, leading to a gradual degradation in system performance. Comparing single and dual spot checks reveals that while both have the same downward trend, the dual spot check consistently maintains a small increase of performance. For example, at $S_i = 7$, TSM improves from 0.9626 to 0.9633 and R climbs from 0.9881 to 0.9884. Overall, these findings highlight that while increasing initial credibility score slightly impacts performance by making it harder to distinguish between colluding and honest agents, the use of a second spot check consistently improves TSM and R across varied settings, emphasizing the value of a dual spot check.

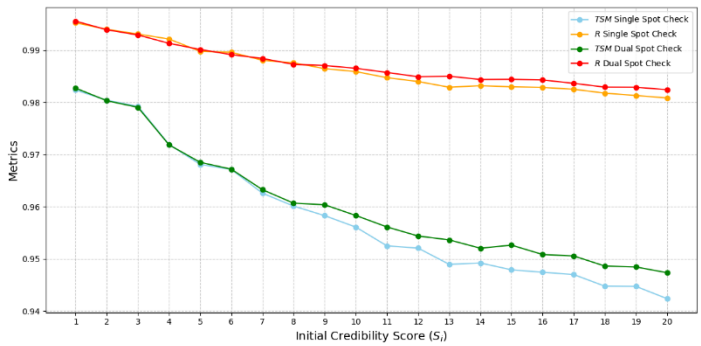

Figure 14. TSM and R under varying initial credibility score.

Table 5. TSM and R under varying cost constraints.

C^*	TSM		R	
	One-stage	Two-stage	One-stage	Two-stage
1	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000
4	0.3490	0.3456	0.3490	0.3456
5	0.7414	0.7504	0.7467	0.7556
6	0.9394	0.9362	0.9448	0.9419
7	0.9662	0.9651	0.9882	0.9876
8	0.9638	0.9651	0.9870	0.9874
9	0.9676	0.9677	0.9885	0.9886
10	0.9678	0.9678	0.9898	0.9899
11	0.9739	0.9744	0.9954	0.9957
12	0.9750	0.9749	0.9961	0.9958
13	0.9770	0.9766	0.9961	0.9960
14	0.9780	0.9776	0.9969	0.9969
15	0.9799	0.9800	0.9977	0.9976

Table 6. TSM and R under varying initial credibility score.

Initial S_i	TSM		R	
	One-stage	Two-stage	One-stage	Two-stage
1	0.9824	0.9827	0.9953	0.9956
2	0.9804	0.9803	0.9940	0.9939
3	0.9793	0.9791	0.9931	0.9929
4	0.9720	0.9719	0.9922	0.9913
5	0.9681	0.9685	0.9898	0.9902
6	0.9672	0.9672	0.9896	0.9892
7	0.9626	0.9633	0.9881	0.9884
8	0.9601	0.9607	0.9876	0.9873
9	0.9583	0.9604	0.9865	0.9871
10	0.9561	0.9583	0.9859	0.9866
11	0.9525	0.9561	0.9848	0.9858
12	0.9521	0.9544	0.9840	0.9849
13	0.9490	0.9537	0.9829	0.9850
14	0.9492	0.9521	0.9832	0.9844
15	0.9479	0.9527	0.9830	0.9844
16	0.9475	0.9509	0.9829	0.9843
17	0.9470	0.9506	0.9825	0.9837
18	0.9448	0.9487	0.9818	0.9829
19	0.9447	0.9485	0.9813	0.9829
20	0.9424	0.9474	0.9809	0.9825

4. Conclusion and Future Work

This paper presents a novel heterogeneous multi-agent collusion defense system that integrates a two-stage spot checking mechanism and a dynamic credibility score-based allocation scheme. Experimental results indicate that the two-stage spot checking approach consistently outperforms the single-stage spot checking, achieving higher performance across a range of collusion scenarios. Impacts of parameters including the honest agent reliability, resource and spotter agent costs, the number of colluding agents, credibility score adjustment combinations, cost constraint, and initial credibility score are examined.

The experimental studies on a pattern recognition VBM-based system reveal that system performance can benefit from high honest agent reliability, low initial credibility scores, large penalties for incorrect outputs, and a large cost constraint. While the cost ratio between spotter agents and resource agents has a moderate impact, better performance is observed when spotters are affordable to allow enough spot checks to

minimize the risk of collusion without limiting task execution. The number of colluding agents highly influences system performance especially when it is high enough to overwhelm the system. Overall, the two-stage spot checking approach consistently outperforms the one-stage approach, particularly as the number of colluding agents and initial credibility scores increase.

The proposed model assumes that spotter agents are perfectly reliable. This limitation will be addressed in future research by modeling spotter agents with both false negative and false positive detections. Future research also includes exploring the utilization of different trust models and voting techniques to understand their impact on the defense system. Future research would also include scaling the system to include a larger number of agents. Another area of research would be to incorporate AI methods such as reinforcement learning for the master agent to actively and adaptively learn to optimize the parameters of the system. These advancements could enhance further the system's reliability, security, and resilience against diverse adversarial threats.

Conflict of Interest
The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors would like to thank the editor and anonymous reviewers for their comments that help improve the quality of this work.

AI Disclosure
The authors declare that no assistance is taken from generative AI to write this article.

References

Aguiar, N., Venkitasubramaniam, P., & Gupta, V. (2022). Data-driven contract design for multi-agent systems with collusion detection. *IEEE Signal Processing Letters*, *29*, 1002-1006. https://doi.org/10.1109/lsp.2022.3163691.

Avizienis, A., Gilley, G.C., Mathur, F.P., Rennels, D.A., Rohr, J.A., & Rubin, D.K. (1971). The STAR (self-testing and repairing) computer: an investigation of the theory and practice of fault-tolerant computer design. *IEEE Transactions on Computers*, *C-20*(11), 1312-1321. https://doi.org/10.1109/t-c.1971.223133.

Bertrand, Q., Duque, J., Calvano, E., & Gidel, G. (2023). Q-learners can provably collude in the iterated prisoner's dilemma. https://doi.org/10.48550/arXiv.2312.08484.

Bonjour, T., Aggarwal, V., & Bhargava, B. (2022). Information theoretic approach to detect collusion in multi-agent games. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence* (pp. 223-232). PMLR. https://proceedings.mlr.press/v180/bonjour22a.html.

Burgard, W., Moors, M., Stachniss, C., & Schneider, F.E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on robotics*, *21*(3), 376-386. https://doi.org/10.1109/tro.2004.839232.

Chaisawat, S., & Vorakulpipat, C. (2020). Fault-tolerant architecture design for blockchain-based electronics voting system. In *2020 17th International Joint Conference on Computer Science and Software Engineering* (pp. 116-121). IEEE. Bangkok, Thailand. https://doi.org/10.1109/jcsse49651.2020.9268264.

Chen, L., & Avizienis, A. (1978). N-version programming: a fault-tolerance approach to reliability of software operation. In *Proceedings of 8th IEEE International Symposium on Fault-Tolerant Computing* (Vol. 1, pp. 3-9). IEEE Computer Society.

Chen, Y., Wang, Z., Min, Y., & Liu, Z. (2025). Decentralized voting-based federated learning framework for lightweight node selection in edge collaborative IoT. *IEEE Internet of Things Journal, 12*(12), 20272-20287.

Gaur, D., Bali, S., Gunasekaran, A., Joshi, N., Chaudhary, K., & Bali, V. (2025). A multi-criteria decision-making framework for blockchain technology adoption in smart healthcare. *International Journal of Mathematical, Engineering and Management Sciences, 10*(5), 1476-1496.

Gogiashvili, J.G., Namicheishvili, O., & Shonia, G. (2000). Optimization of weights for threshold redundancy of binary channels by the method of (Mahalanobis') generalized distance. In *Second International Conference on Mathematical Methods in Reliability: Methodology, Practice and Interference* (pp. 4-7). https://scholar.google.com/citations?view_op=view_citation&hl=en&user=4mromi8aaaaj&citation_for_view=4mromi8aaaaj:_fxgofyzp5qc.

Iqbal, S., Qureshi, A.N., Alhussein, M., Aurangzeb, K., Mahmood, A., & Azzuhri, S.R.B. (2025). Dynamic selectout and voting-based federated learning for enhanced medical image analysis. *Machine Learning: Science and Technology*, *6*(1), 015010. https://doi.org/10.1088/2632-2153/ada0a6.

Ivanov, R., Pajic, M., & Lee, I. (2016). Attack-resilient sensor fusion for safety-critical cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, *15*(1), 1-24. https://doi.org/10.1145/2847418.

Lei, T., Zhang, Q., Qiu, W., Zheng, H., Miao, S., Jie, W., Zhu, J., Dong, J., & Zheng, Z. (2025). An enhanced DPoS consensus mechanism using quadratic voting in Web 3.0 ecosystem. *Blockchain, 3*(1), 1-14. https://doi.org/10.55092/blockchain20250001.

Levitin, G., Xing, L., & Dai, Y. (2017). Optimal spot-checking for collusion tolerance in computer grids. *IEEE Transactions on Dependable and Secure Computing*, *16*(2), 301-312. https://doi.org/10.1109/tdsc.2017.2690293.

Levitin, G., Xing, L., Johnson, B.W., & Dai, Y. (2018). Optimization of dynamic spot-checking for collusion tolerance in grid computing. *Future Generation Computer Systems*, *86*, 30-38. https://doi.org/10.1016/j.future.2018.01.049.

Li, H., Hui, H., & Zhang, H. (2023). Decentralized energy management of microgrid based on blockchain-empowered consensus algorithm with collusion prevention. *IEEE Transactions on Sustainable Energy*, *14*(4), 2260-2273. https://doi.org/10.1109/tste.2023.3258452.

Miao, B., Xu, Y., Wang, J., & Zhang, Y. (2024). Dc-bvm: dual-channel information fusion network based on voting mechanism. *Biomedical Signal Processing and Control*, *94*, 106248. https://doi.org/10.1016/j.bspc.2024.106248.

Motwani, S.R., Baranchuk, M., Strohmeier, M., Bolina, V., Torr, P.H.S., Hammond, L., & de Witt, C.S. (2024). Secret collusion among ai agents: multi-agent deception via steganography. *Advances in Neural Information Processing Systems*, *37*, 73439-73486. https://proceedings.neurips.cc/paper_files/paper/2024/file/861f7dad098aec1c3560fb7add468d41-Paper-Conference.pdf.

Nguyen, G.T., & Kim, K. (2018). A survey about consensus algorithms used in blockchain. *Journal of Information Processing Systems*, *14*(1), 101-128. https://doi.org/10.3745/jips.01.0024.

Nordmann, L., & Pham, H. (1999). Weighted voting systems. *IEEE Transactions on Reliability*, *48*(1), 42-49. https://doi.org/10.1109/24.765926.

Owoputi, R., & Ray, S. (2022). Security of multi-agent cyber-physical systems: a survey. *IEEE Access*, *10*, 121465-121479. https://doi.org/10.1109/access.2022.3223362.

Parhami, B. (1994). Voting algorithms. *IEEE Transactions on Reliability*, *43*(4), 617-629. https://doi.org/10.1109/24.370218.

Pilar, B. (2025). InceptionV3-driven multiclassifier voting system for watermark classification. In *2025 International Conference on Computational, Communication and Information Technology* (pp. 471-476). IEEE. Indore, India. https://doi.org/10.1109/icccit62592.2025.10928146.

Staab, E., & Engel, T. (2009). Collusion detection for grid computing. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (pp. 412-419). IEEE. Shanghai, China. https://doi.org/10.1109/ccgrid.2009.12.

Tasci, E., Uluturk, C., & Ugur, A. (2021). A voting-based ensemble deep learning method focusing on image augmentation and preprocessing variations for tuberculosis detection. *Neural Computing and Applications*, *33*(22), 15541-15555. https://doi.org/10.1007/s00521-021-06177-2.

Tassa, T., Grinshpoun, T., & Yanai, A. (2019). A privacy preserving collusion secure DCOP algorithm. *Cryptography and Security.* https://doi.org/10.48550/arXiv.1905.09013.

Tham, C.K., & Renaud, J.C. (2005). Multi-agent systems on sensor networks: a distributed reinforcement learning approach. In *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (pp. 423-429). IEEE. Melbourne, VIC, Australia. https://doi.org/10.1109/issnip.2005.1595616.

Verma, S., Chandra, G., & Yadav, D. (2025). LVCA: an efficient voting-based consensus algorithm in private blockchain for enhancing data security. *Peer-to-Peer Networking and Applications*, *18*(2), 88. https://doi.org/10.1007/s12083-024-01808-6.

Wang, S., Qu, X., Hu, Q., Wang, X., & Cheng, X. (2023). An uncertainty-and collusion-proof voting consensus mechanism in blockchain. *IEEE/ACM Transactions on Networking*, *31*(5), 2376-2388. https://doi.org/10.1109/tnet.2023.3249206.

Wang, W., An, B., & Jiang, Y. (2018). Optimal spot-checking for improving evaluation accuracy of peer grading systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1), 833-840. https://doi.org/10.1609/aaai.v32i1.11336.

Wang, W., An, B., & Jiang, Y. (2020). Optimal spot-checking for improving the evaluation quality of crowdsourcing: application to peer grading systems. *IEEE Transactions on Computational Social Systems*, *7*(4), 940-955. https://doi.org/10.1109/tcss.2020.2998732.

Wu, J., Xu, X., Zhang, P., & Liu, C. (2011). A novel multi-agent reinforcement learning approach for job scheduling in grid computing. *Future Generation Computer Systems*, *27*(5), 430-439. https://doi.org/10.1016/j.future.2010.10.009.

Zhang, H., & Zhou, Y. (2023). A neural network-based weighted voting algorithm for multi-target classification in WSN. *Sensors*, *24*(1), 123. https://doi.org/10.3390/s24010123.