A Survey of LLM-Driven AI Agent Communication: Protocols, Security Risks, and Defense Countermeasures

Dezhang Kong, Shi Lin, Zhenhua Xu, Zhebo Wang, Minghao Li, Yufeng Li, Yilun Zhang, Zeyang Sha, Yuyuan Li, Changting Lin, Xun Wang, Xuan Liu, Muhammad Khurram Khan, Ningyu Zhang, Chaochao Chen, Meng Han

arXiv:2506.19676v1 [cs.CR] 24 Jun 2025

***Abstract*—In recent years, Large-Language-Model-driven AI agents have exhibited unprecedented intelligence, flexibility, and adaptability, and are rapidly changing human production and lifestyle. Nowadays, agents are undergoing a new round of evolution. They no longer act as an isolated island like LLMs. Instead, they start to communicate with diverse external entities, such as other agents and tools, to collectively perform more complex tasks. Under this trend, *agent communication* is regarded as a foundational pillar of the future AI ecosystem, and many organizations intensively begin to design related communication protocols (e.g., Anthropic's MCP and Google's A2A) within the recent few months. However, this new field exposes significant security hazard, which can cause severe damage to real-world scenarios. To help researchers to quickly figure out this promising topic and benefit the future agent communication development, this paper presents a comprehensive survey of *agent communication security*. More precisely, we first present a clear definition of agent communication and categorize the entire lifecyle of agent communication into three stages: user-agent interaction, agent-agent communication, and agent-environment communication. Next, for each communication phase, we dissect related protocols and analyze its security risks according to the communication characteristics. Then, we summarize and outlook on the possible defense countermeasures for each risk. Finally, we discuss open issues and future directions in this promising research field.**

***Index Terms*—large language model, AI agent, agent communication, attack and security**

I. Introduction

The emergence of Large Language Models (LLMs) has led to revolutionary advancements in Artificial Intelligence (AI), exhibiting unprecedented capabilities in understanding complex tasks [308]. More importantly, LLMs greatly boosted the ideal form of AI that human expects: ***agents***[1]. Different from LLMs that are mainly like chat bots, agents possess more *comprehensive* capabilities (e.g., perception, interaction, reasoning, and execution), enabling them to *independently* complete a real-world task. For example, when users seek to make a travel plan, LLMs can only provide the best plan *in text*, while agents can realize the best plan *in action*, such as checking the weather, buying tickets, and booking hotels. Agents greatly speed up the progress of the intelligence transformations of enterprise. Their market size is respected to increase by 46% per year [222]. It can be foreseen that agents will subvert the production and living patterns of modern society, greatly changing the future business landscape. As a result, developing and promoting agents have become the strategic planning of major powers and influential companies.

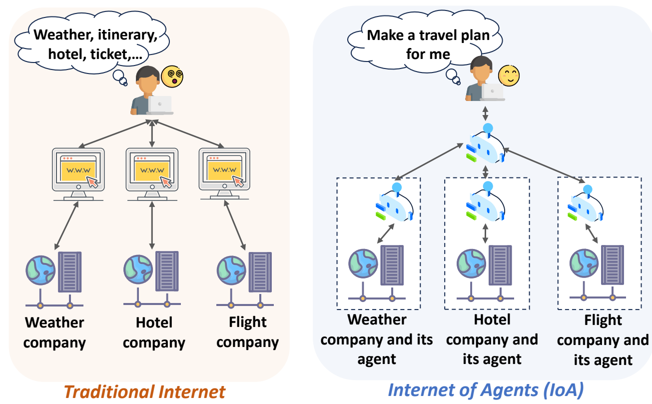

Fig. 1. The comparison between traditional Internet and Internet of Agents (IoA). In traditional Internet, users need to manually visit different websites to finish a travel, which is cumbersome. With Internet of Agents, users only need to assign a task to their agent, which will communicate with the agents of different companies (e.g., hotel and train companies) to automatically finish a best travel plan.

Now, agents are evolving towards the direction of *domain-specific* entities, i.e., being customized for specific scenarios and tasks. In this context, as shown in Figure 1, a task usually requires the collaboration of multiple agents, which may locate globally on the Internet. Under this condition, ***agent communication*** becomes the foundation of the future AI ecosystem. It enable agents to find other agents with specific capabilities, access external knowledge, assign tasks, and achieve other interactions. Based on the vast market of agent communication, an increasing number of communities and companies are seizing the opportunity to contribute to the development of agent communication. In November, 2024, Anthropic proposed Model Context Protocol (MCP) [16], a universal protocol that allows agents to invoke external environments, such as datasets, tools, and APIs. MCP quickly gained a great deal of attention in the recent few months. Up to now, hundreds of enterprises have announced their access

Dezhang Kong, Zhenhua Xu, Zhebo Wang, Changting Lin, Ningyu Zhang, Chaochao Chen, and Meng Han are with Zhejiang University, Hangzhou, China (email: kdz@zju.edu.cn, xuzhenhua0326@zju.edu.cn, breynald@zju.edu.cn, lct@gentel.com, zhangningyu@zju.edu.cn, zjuccc@zju.edu.cn, mhan@zju.edu.cn); Shi Lin and Xun Wang are with Zhejiang Gongshang University, Hangzhou, China (email: linshizjgsu@gmail.com, wx@zjgsu.edu.cn); Minghao Li is with Heilongjiang University, Harbin, China (email: mhli@s.hlju.edu.cn); Yufeng Li is with East China Normal University (email: liyufeng2187@163.com); Yilun Zhang is with Purdue University, West Lafayette, US (email: zhan4984@purdue.edu); Zeyang Sha is with Ant Group, Hangzhou, China (email: shazeyang.szy@antgroup.com); Yuyuan Li is with Hangzhou Dianzi University, Hangzhou, China (email:y2li@hdu.edu.cn); Xuan Liu is with Yangzhou University, Yangzhou, China (email: yusuf@yzu.edu.cn); Muhammad Khurram Khan is with King Saud University, Riyadh, Saudi Arabia (email: mkhurram@ksu.edu.sa). Changting Lin and Meng Han are also with GenTel.io, Hangzhou, China.

[1]In this paper, all agents refer to LLM-driven AI agents.

to MCP, such as OpenAI [203], Google [87], Microsoft [53], Amazon [21], Alibaba [10], and Tencent [251], and MCP's package receives over 3 million weekly downloads [17]. In April 2025, Google proposed Agent to Agent Protocol (A2A) [218], which enables seamless communication and collaboration among agents. Since its release, A2A has received extensive supports from many enterprises, such as Microsoft [188], Atlassian [149], and PayPal [229]. It can be seen that the breakthroughs in agent communication are bringing rapid and profound changes and will become an indispensable part of the AI ecosystem.

However, the rapid development of agent communication also introduce complex security risks that could cause severe damage in the AI ecosystem. For example, the collaboration of cross-organization agents significantly enlarging the attack surface, causing severe security risks, including but not limited to privacy leakage, agent spoofing, agent bullying, and Denial of Service attacks. Since the research related to agent communication is still *in the nascent stage*, it urgently needs a systematic review of the security problems existing in the complete agent communication lifecycle. Following this trend, *this paper aims to provide a comprehensive survey of existing agent communication techniques, analyze their security risks, and discuss possible defense countermeasures.* We believe this work can help a broad range of readers, such as researchers who devote to agent development and beginners who just start their journey in AI.

The contributions of this paper are as follows:

- We present the first systematic overview of agent communication. Specifically, we propose a definition of agent communication for the first time, and classify it into three stages based on the communication object: user-agent interaction, agent-agent communication, and agent-environment communication. This classification covers the entire lifecycle of agent communication, and the protocols in the same stage exhibit similar attack surfaces, which can help future studies to analyze and evaluate their work more conveniently.
- We make in-depth analyses of the security risks in the evolvement of agent communication, discussing the discovered attacks and those that have yet not been revealed. Our analyses show that user-agent interaction mainly suffers from threats from faulty and malicious user inputs, agent-agent communication faces significant attacks from other agents and in-the-middle adversaries, and agent-environment communication can be easily impacted by compromised external tools and resources.
- We detailedly discuss the targeted defense countermeasures for mitigating the exposed security risks, pointing out the possible development directions. For user-agent interaction, effective filter of multimodal inputs is necessary; agent-agent communication needs powerful mechanisms to monitor, archive, audit, and quantify the responsibility of actions in agent collaboration; agent-environment communication should be protected by powerful detection of poisoned external environments.
- We finally discuss the open issues and future research directions. We not only point out the much needed techniques that can defend agent communication, but also explained that related laws and regulations should be improved as soon as possible. Only when both aspects develop simultaneously can the security of agent communication in practice gets guaranteed.

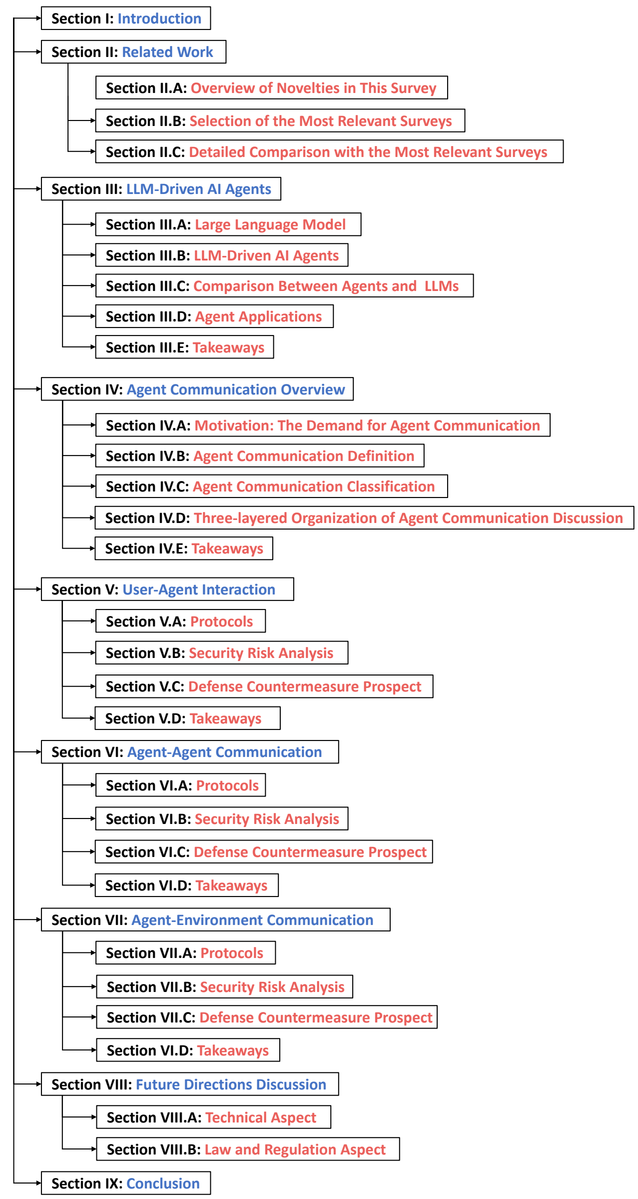

Fig. 2. The organization of this survey.

Organization. As shown in Figure 2, we organize this survey as follows. Section II compares the most relevant surveys with this paper and outlines the novelties in this survey. Section III introduces the preliminaries of this survey. Section IV presents a definition and classification of agent communication. Section V introduces user-agent interaction protocols and analyzes related security risks and defense countermeasures. Section VI exhibits agent-agent communication protocols, related security risks, and corresponding defense countermeasures. Similarly, Section VII shows the protocols, risks, and defenses for user-environment communication. In Section VIII, we discuss the

TABLE I

COMPARISON BETWEEN DIFFERENT SURVEYS, WHERE "MOTI." REFERS TO THE MOTIVATION OF PROPOSING AGENT COMMUNICATION; "DEFI." REFERS TO THE DEFINITION OF AGENT COMMUNICATION; "CLAS." REFERS TO THE CLASSIFICATION OF AGENT COMMUNICATION OR PROTOCOLS; "U-A" REFERS TO USER-AGENT INTERACTION; "A-A" REFERS TO AGENT-AGENT COMMUNICATION; "A-E" REFERS TO AGENT-ENVIRONMENT COMMUNICATION; "SECU." REFERS TO SECURITY; "COMM." REFERS TO COMMUNICATION; "RESEARCH OBJECT" DENOTES THE THEME OF A SURVEY; "AGENT COMMUNICATION" DENOTES WHETHER A SURVEY CONCENTRATE ON AGENT COMMUNICATION; "PROTOCOL COVERAGE" DENOTES WHETHER A SURVEY INCLUDES COMPREHENSIVE AGENT COMMUNICATION PROTOCOLS; "SECURITY ANALYSES" DENOTES WHETHER A SURVEY ANALYZES THE SECURITY RISKS OF DIFFERENT AGENT COMMUNICATION STAGES; "DEFENSE PROSPECT" DENOTES WHETHER A SURVEY ANALYZES THE POSSIBLE DEFENSES FOR DIFFERENT AGENT COMMUNICATION STAGE; "RELE." REFERS TO THE DEGREE OF RELEVANCE BETWEEN A SURVEY AND THIS SURVEY, WHERE THE HIGHER THE SCORE, THE MORE RELEVANT IT IS; ✗: NOT DISCUSSED IN THIS SURVEY; ✓: MENTIONED BUT NOT A MAIN FOCUS OR NOT DISCUSSED COMPREHENSIVELY IN THIS SURVEY; ✔: COMPREHENSIVELY DISCUSSED IN THIS SURVEY.

Survey	Year	Research Object	Rele.	Agent Communication			Protocol Coverage				Security Analyses			Defense Prospect		
				Moti.	Defi.	Clas.	Clas.	U-A	A-A	A-E	U-A	A-A	A-E	U-A	A-A	A-E
[161]	2024	Personal Agent	4	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗
[72]	2024	Agent Secu.	5	✗	✗	✗	✗	✗	✗	✗	✔	✗	✓	✔	✗	✓
[153]	2025	Agent Secu.	5	✗	✗	✗	✗	✗	✗	✗	✔	✓	✗	✔	✓	✗
[133]	2025	Agent Secu.	5	✗	✗	✗	✗	✗	✗	✗	✓	✔	✓	✓	✔	✓
[52]	2025	Agent Secu.	5	✗	✗	✗	✗	✗	✗	✗	✔	✓	✓	✔	✓	✓
[98]	2025	Agent Secu.	5	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✓
[276]	2025	General IoA	5	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✓	✓
[93]	2024	Agent Secu.	5	✗	✗	✗	✗	✗	✗	✗	✔	✗	✓	✔	✗	✓
[298]	2025	Agent Comm.	6	✗	✗	✔	✔	✗	✗	✗	✓	✓	✓	✓	✓	✓
[277]	2025	Agent Secu.	6	✗	✗	✗	✗	✗	✓	✓	✓	✔	✔	✓	✔	✔
[263]	2025	General Agent	6	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
[278]	2024	Agent Secu.	6	✔	✗	✗	✗	✗	✗	✗	✔	✗	✓	✔	✗	✓
[69]	2025	General Agent	6	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗
[228]	2025	Agent Comm.	7	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗
[306]	2025	Agent Secu.	7	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
[86]	2025	Agent Secu.	7	✗	✗	✗	✗	✗	✗	✗	✗	✔	✗	✗	✔	✗
[299]	2025	Agent Comm.	8	✔	✗	✔	✔	✓	✓	✓	✗	✗	✗	✗	✓	✓
[137]	2025	Agent Comm.	8	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓
[239]	2025	Agent Comm.	8	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓
[220]	2025	Agent Comm.	8	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗
[62]	2025	Agent Comm.	8	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✓	✓
[101]	2025	MCP Security	8	✗	✗	✗	✗	✗	✗	✓	✗	✗	✔	✗	✗	✔
This survey		Agent Comm. Secu.	/	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

open issues and future research direction. Section IX concludes this survey.

II. RELATED WORK

A. Overview of Novelties in This Survey

Table I summarizes the characteristics of the most relevant surveys and the differences between this survey and previous surveys. In summary, this surveys exhibit the following novelties:

- This survey presents a comprehensive illustration of agent communication. Specifically, it explains why the current agent ecosystem needs communication (i.e., the predicament faced by a single agent, Section IV-A), gives the definition of agent communication (Section IV-B), and proposes a novel classification principle based on communication entity, which can cover the entire lifecycle of agent communication (Section IV-C). As a result, future studies can be included and categorized according to our survey seamlessly, which benefits the systematic study and development of this field.
- This survey exhibits a comprehensive illustration of the existing protocols related to different agent communication stages (Sections V-A, VI-A, and VII-A), including newly proposed and previously neglected protocols that have not been discussed by other surveys. Besides, we categorize these protocols based on their architecture and summarize corresponding characteristics, rather than mechanically listing each protocol. This organization method can allow any researchers interested in this field quickly establish a preliminary but comprehensive understanding of agent communication.
- This survey makes an in-depth analysis of the found and potential security risks that have not been revealed for each agent communication stage (Sections V-B, VI-B, and VII-B). We introduce both the vulnerability from which attacks derive and typical attack examples, systematically pointing out the drawbacks of the current agent communication. To our knowledge, there have not been studies mainly focusing on the security risks of agent communication.
- This survey throughly outlooks on the possible defense countermeasures (Sections V-C, VI-C, and VII-C) that can make future agent communication more secure and reliable, covering the whole lifecycle of agent communication, which is not achieved by previous surveys.

B. Selection Principles of the Most Relevant Surveys

Challenge. Our survey aims at comprehensively studying the protocols, related security risks, and possible defenses of agent communication. However, *there are a lot of surveys that seem relevant but are actually different in essence. Listing these surveys is not conducive to readers' understanding of this field as efficiently as possible, especially for those who want to read the original texts of these surveys.*

To solve this challenge, when selecting the most relevant surveys, we focus on three principles: LLM-driven agents, agent communication, and security. However, to our knowledge, there have not been papers systematically discuss all of these three themes. As a result, **as long as a survey meets two of the three principles, we will treat it as a relevant survey.**

- **Principle #1: LLM-driven agents**. The first and the most important is that the research object of a survey must be LLM-driven agents. *This principle must be satisfied*. This is because there have been many studies about multi-agent systems (MAS) before the emergence of LLMs. These agents have completely different cores and characteristics from LLM-driven agents, so discussing them benefits very little to this survey. Besides, surveys focusing on only LLMs instead of LLM-driven agents are also not listed in Table I (but we will draw on their valuable insights in other sections). This is because LLMs shows significant difference from agents, which we make a detailed illustration in Section III-C. As a result, researching LLM-driven agents is the most important principle.
- **Principle #2: agent communication.** The second principle is that a survey focus on or partially discuss agent communication, especially including some typical agent communication protocols such as MCP. This is because agent communication is very different from agent. However, if a survey satisfies the other two principles (i.e., LLM-driven agents and security), we still treat it as a relevant survey.
- **Principle #3: security.** The final principle is that a survey focus or partially discussed agent-related security. This is because we believe that the security risks of agents still have meaning to the security risks of agent communication. The former is usually a subset of the latter, i.e., agent communication shows novel and more attack surfaces compared to agent.

Relevance Score. As a result, we can find that there are two main types of relevant surveys: LLM-driven agents + communication, or LLM-driven agents + security. As shown in Table I, we list a relevance score for each survey. The higher the score, the more relevant we think it is to our survey. This score is subjectively derived by us after carefully reading the paper and does not have an objective calculation method. This is because we found that the forms of surveys are highly diverse, and it is hard to accurately classify them using merely several metrics. As a result, we directly present the score based on our subjective feelings when reading these surveys.

C. Detailed Comparison with the Most Relevant Surveys

In this section, we will detailedly compare the most relevant surveys in Table I with our survey.

Survey [161] focuses on personal agents that deeply integrate personal data and devices, exploring their potential as the main software paradigm for future personal computer. It only partially mentions the security risks related to personal agents in a section. Besides, these risks only belongs to the user-agent interaction phase. It also does not discuss agent communication and related security risks and defenses. Survey [72] focuses on agent security instead of agent communication security. It is a security-specific paper, so its discussion of security is more comprehensive compared to [161]. However, the main body of its discussion is about the interaction between user and agent (U-A), without enough considerations about agent-agent (A-A) or agent-environment (A-E) interaction, which have significantly different characteristics. Besides, it also does not include any protocols related to agent communication. Survey [153] also focuses on agent security instead of agent communication security, which is similar to [72]. This survey focuses on single-agent systems and partially discussed multi-agent collaboration. It does not consider agent communication, related protocols, and enough security analyses about A-A and A-E. Survey [133] systematically summarizes seven security challenges for multi-agent systems. As shown in Table I, its main focus in on A-A, and only partially discusses U-A and A-E, which is not comprehensive. Besides, it does not consider agent communication and related protocols. Survey [52] propose four knowledge gaps faced by agents, which mainly fall within U-A, partially discussing A-A and A-E. Besides, it does not consider agent communication or any related protocols. The defense prospect is also limited. Survey [98] focuses on the security risks of U-A and A-E, such as malicious API. It does not consider agent communication and related protocols. Besides, its security analyses are also not comprehensive enough. Survey [276] focus on the fundamentals, applications, and challenges of IoA. Since its focus is different, agent communication and related security are only partially mentioned. Specifically, it only introduces a few related protocols and briefly makes an analysis about related security. Besides, it also lacks the related illustration (such as definition and classification) of agent communication. Survey [93] also concentrates on the security of U-A, partially discussing A-E. It does not mention agent communication and related protocols, as well as the risks of A-A. Survey [298] focus on agent communication architecture, which is a study of agent interaction mechanisms from a high-level and abstract view. Besides, it only partially mentioned related security, and did not discuss any communication protocols. Survey [277] focuses on the security of IoA. It mentioned a few agent communication protocols (i.e., MCP, A2A, ANP, and Agora), neglecting many other important protocols. Besides, it lacks the motivation, definition, and classification of agent communication, and also does not classify protocols. According to our analyses, the security analyses (especially for U-A) are also not comprehensive enough. Survey [263] propose the concept of "full stack safety" of agents, providing comprehensive analyses of data preparation, pre-training, post-training, deployment, and commercialization. It does not focus on agent communication security. As a result, this survey did not give a clear illustration of agent communication, only mentioned a few protocols (i.e., MCP, A2A, ANP, and Agora), and partially discussed related threats and countermeasures. Survey [278] gives a comprehensive analyses of the security of agent networks. However, it does not include the discussion of communication protocols, and lacks enough security analyses of A-A and A-E. Survey [69] does not focus on security. Instead, it concentrates on evaluating LLMs and agents. Besides, it also analyzes the architecture of some communication protocols (i.e., MCP, A2A, and ANP). We can see that it does not give a detailed illustration of agent communication, enough coverage of protocols, or comprehensive discussion about security. Survey [228] focuses on MCP, detailedly analyzed related architectures and applications. It does not consider other communication protocols and only partially mentioned security-related contents. Survey [306] analyzed the threats of agents and dividing them into two categories (intrinsic and extrinsic), partially covering U-A, A-A, and A-E. However, its analyses are not comprehensive enough, and it did not mention any communication protocols. Survey [86] makes a comprehensive analyses of the risks for multi-agent systems. However, its focus only fall within A-A, not considering U-A, A-E, and related protocols.

Survey [299] is one of the surveys with highest relevance scores because it focuses on agent communication and analyzes related protocols. However it still have significant differences from our survey. First, it lacks some critical protocols like AG-UI, ACP-AgentUnion, ACN, Agent Protocol, API Bridge Agent, and Function Calling. Second, it lacks analyses of security threats. Third, its defenses prospect is limited. Survey [137] focuses on the influences of MCP. As a result, it lacks other protocols, the illustration of agent communication, and security-related discussion. Similarly, survey [239] also focuses on MCP. It lacks illustrations of other protocols and comprehensive security analyses. Survey [220] comprehensively introduces A2A, lacking discussion about other protocols and security analyses. Survey [62] detailed discussed MCP, A2A, ANP, and ACP(-IBM). It also partially analyzed related security risks and defenses. However, there still lacks other protocols, in-depth security analyses, and systematic illustration of agent communication. Hou et al. [101] discussed the security risks of MCP. They did not consider other protocols and the high-level overview of agent communication.

III. LLM-Driven AI Agents

In this section, we review the entire lifetime from LLM to LLM-driven AI agent. Our goal is to help beginners to quickly figure out agents, their characteristics, relationships, and applications.

A. Large Language Model

Large Language Model (LLM) is a new type of artificial intelligence (AI) model trained on large-scale text corpora to understand and generate human language [200]. Once it came out, LLMs have demonstrated unprecedented capabilities across a wide range of domains, including but not limited to natural language understanding and generating [311], logic reasoning [212], [281], [339], code generation [325], and translation [209]. These remarkable performances can be attributed to two major factors. One is that LLMs are built upon a powerful architecture known as the Transformer [258], which effectively models and captures contextual dependencies between tokens and dynamically weighs the importance of different parts of the input. The other key factor, perhaps the most important one, is the *massive scales* of LLMs that far exceeds traditional AI models. When model parameters surpass certain thresholds, LLMs exhibit *emergent abilities* [280], referring to unexpected capabilities that do not appear in smaller models. As shown in Table II, the parameter scale of a LLM can be hundreds or thousands of times that of traditional AI models.

TABLE II
Comparison of Model Architectures and Parameter Scales.

Architecture	Model	Year	Parameters
FNN	MLP	1990s	100K
FNN	LeNet-5 [146]	1998	60K
RNN	Elman Net [64]	1990	100K
LSTM	LSTM [99]	1997	1–10M
CNN	ResNet-50 [94]	2015	25M
CNN	AlexNet [138]	2012	60M
CNN	VGG-16 [238]	2014	138M
GAN	DCGAN [215]	2016	4M
GNN	GCN [132]	2017	23K
Autoencoders	DAE [259]	2008	100K
Autoencoders	VAE [131]	2013	1M
Transformer	GPT-3 [25]	2020	175B
Transformer	PaLM [39]	2022	540B
Transformer	GPT-4 [4]	2023	1T
Transformer	DeepSeek-V3 [165]	2024	671B
Transformer	DeepSeek-R1 [85]	2025	671B

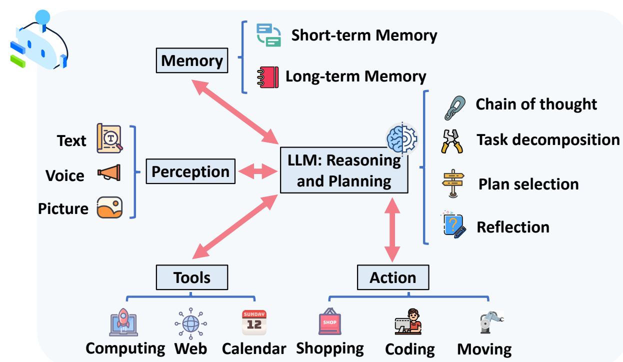

Fig. 3. A typical architecture of LLM-driven agents.

B. LLM-Driven AI Agents

Figure 3 illustrates a typical architecture of LLM-driven agents. Different from LLMs that mainly act as chat bots and do not possess professional ability in specific domains, agents are designed to automatically help human to finish specialized tasks. To this end, agents are equipped with multiple modules to become more all-powerful. As shown in Figure 3, there are usually five modules in agents: perception, memory, tools, reasoning, and action.

- **Perception module.** To automatically finish a specified task, agents need the ability to perceive the real-world environment. For example, the autonomous driving agent needs to sense road conditions in real time so as to take actions such as avoiding, driving, or braking [179], [183]. The type of perception ability depends on the domain for which the agent is designed. For instance, an autonomous driving agent need the ability of visual or radar perception [244], [302], while a code-generating agent may not require such functions [106], [114].
- **Memory module.** The processing of real-world tasks also require the strong ability of memory. Agents need to have long-term memory to store complex instructions, knowledge, environment interaction history, or other data that may be required in future steps [92], [180], [309]. This usually require external storage resources to assist the brain, such as database or memory sharing [75], [77]. In contrast, LLMs do not have such an excellent memory ability. Their memory is short-term, which only lasts for rounds of conversations [266], [340].
- **Reasoning and planing module.** LLM acts as the brain of agents due to its excellent ability of reasoning and planing. It intercepts the instructions from users and automatically decompose the received task into multiple feasible steps [117], [129], [240], [273]. Then, it selects a best plan from different candidates [103], [126], [336]. Besides, it also revises strategies based on environmental feedback, mitigating errors like code bugs or logical inconsistencies [221], [257], [286], [344]. For example, when the autonomous driving module finds that the barrier is closer, it will changes the plan to slow down or detour.
- **Tool module.** The tool module is responsible for deeply integrating external resources with the cognitive capabilities of the agent, enabling it to perform complex operations beyond the native capabilities of LLM [158], [175], [285], [312]. For example, through predefined functional interfaces and protocols, a math agent is able to invoke the external computation libraries and symbolic solvers to help it solve mathematical problems [82].
- **Action module.** The action module is the core hub for the interaction with the environment. It is responsible for converting the decisions made by LLMs into executable physical or digital operations and obtaining feedback [272], [326]. This module ensures the executability of instructions through structured output control. For example, it immediately stops generating when LLM generate a complete action description to avoid redundant output interfering with subsequent parsing.

By integrating the above modules, agents establishes a closed-loop system that achieves a full chain of *perception-decision-action-feedback*. As a result, agents achieve unprecedented ability in automatically finishing domain-specific tasks, being closer to the ultimate form of AI that human expects.

TABLE III
COMPARISON BETWEEN AGENTS AND LLMS. [pink] MEANS WORSE, WHILE [blue] MEANS BETTER.

Metric	LLM	Agent
Autonomy	Prompt-dependent	Autonomous
Multimodal interaction	Limited	Strong
Tool Invocation	Simple API	Various tools
Hallucination inhibition	Weak	Strong
Dynamic adaptability	Limited	Strong
Collaboration ability	Limited	Strong
Security	Better	Worse

C. Comparison Between Agents and LLMs

Table III illustrates the advantages of agents over LLMs on different metrics. Overall, agents have many advantages over LLMs except security.

- **High Autonomy.** LLMs can only passively react to the user prompts and then generate responses. They are unable to plan or execute tasks independently. Besides, the response quality highly rely on the prompt skill [29], [63], [80], [167], [191], [283], [343], which seriously affects the user experience. In contrast, agents possess independent capabilities for task decomposition, strategy adjustment, and external tool invocation, which breaks through the passive mode of LLMS and is highly autonomous.
- **Flexible Multimodal interaction.** LLMs have limited capability of handling multimodal inputs, such as text and pictures [144], [227], [319], [323], [343], [346]. Besides, their outputs are also mainly single-modal (e.g., text-only or picture-only), lacking the ability to actively invoke tools to perform physical actions or generate multimodal content. In contrast, agents overcome this drawbacks by deploying multimodal perception frameworks and tool invocation interfaces. They can realize interactions with complex environments, including vision, text, voice, and other physical elements.
- **Abundant Tool invocation.** LLMS usually passively invoke a single tool (such as Function Calling [202]) through predefined API interfaces and can only perform fixed operations as instructed (e.g., calling the weather API to answer queries [318]). In contrast, agents have the ability of active decision-making. They can independently select, combine and dynamically adjust multiple tools, such as connecting crawlers, databases, and visualization tools, to generate responses [97].
- **Better Hallucination inhibition.** LLMs suffer from a serious problem called *hallucination*, which refers to that LLMs are likely to generate non-existent knowledge [84], [108], [160], [256], [296], [329]. LLMs mainly rely on the knowledge internalization of training data, making them prone to hallucinations when facing uncovered domains or outdated information. In contrast, agents are able to reduce the error rate by integrating multiple techniques

such as Retrieval Augmentation Generation (RAG) [79], [151], [335] or other methods, which can align the action of agents [74], [252].

- **Dynamic adaptability.** Essentially, LLMs are static models whose knowledge is fixed at the training phase. Although techniques such as fine-tuning [105], [164], [315] or model editing [162], [265], [270], [303], [322] reduce the training cost significantly, LLMs still cannot adapt to real-time events well. In contrast, agents are equipped with techniques like on-line web search, database query, or real-time sensors, which make them able to dynamically adapt to the changes of real-time environments and information.
- **Stronger Collaboration ability.** LLMs lack enough collaboration ability when handling complex tasks. First, LLMs cannot interact with tools well, they can only access limited external assistance via simple API. Second, different LLMs lack effective cooperation mechanisms. In contrast, agents have designs for multi-agent collaboration. For example, MCP enables agents to use unified integration of external tools, and A2A allows agents from different enterprises to cooperatively finish a task.
- **Worse Security.** Agents have WORSE security than LLMs, which is a major weakness of agents. This is because LLMs are only capable of outputting text. Even if the outputs contain illegal or discriminatory contents, their influences to the real world are limited. In contrast, since agents are endowed with the ability to invoke tools, they can cause *substantial* damages to the real world, including but not limited to maliciously/wrongly operating machines, poisoning databases, and paralyzing the system. As a result, it is necessary to concentrating more on the security of agents.

D. Agent Applications

Due to the strong advantages that agents have shown, related applications are booming. They span multiple domains, from scientific research to engineering systems and social services. Since the application of agents is not the focus of this paper, we will present an brief overview of their practical use cases to illustrate the rapid popularization of agent.

Scientific Research. Agents are increasingly embedded into the research workflow, enhancing ideation, automation, and discovery. Their contributions span multiple disciplines, such as mathematics [51], [150], [267], [293], chemistry [23], [38], [46], [226], biological sciences [166], [290], [294], and materials Science [140], [185], [208]

Technical and Engineering Systems. Agents play a growing role in engineering domains, improving automation, system and software intelligence. For example, agents are widely used in software engineering, assisting in code generation, bug localization, verification, and system configuration [33], [104], [115], [128], [176], [262]. Besides, agents are also popular in game development and simulation [189], [241]. Embodied intelligence is also another hot topic [32], [182].

Social Governance and Public Services Agent are increasingly deployed in sectors focused on public service and human welfare. For example, agents are now widely used in the legal field to help draft contracts, review legal documents, check compliance rules, and analyze cases [111], [187], [253], [254]. Besides, other fields, such as financial services [67], [88], [89], [155], [201], [300], [310], education [55], [59], [181], [194], [245], [269], and healthcare [19], [31], [68], [113], [148], [225], [271], [279], [327], are also actively integrating agents into their respective practices.

Overall, it can be seen that agents are being widely applied in all walks of life, greatly promoting the development of productivity. More importantly, the application of agents is still in its infancy and has an even greater space for development in the future. It is estimated that the agent market will grow at a rate of 40% annually and is expected to exceed 216.8 billion dollars by 2035 [14].

E. Takeaways

Agents show multiple advantages over LLMs on multiple metrics, such as richer perception ability, stronger learning ability, and higher adaptivity. Now, agents are quickly deployed in various domains, providing unprecedented assistance to different groups. Especially, to improve the service quality, agents are evolving towards refinement to obtain professional skill in a small domain, no longer pursuing the comprehensive capabilities like LLM. In contrast, LLMs are more like an intermediate transitional form of the future intelligence, while agents are the next stage development direction of artificial intelligence. It can be foreseen that they will ultimately become indispensable components of future production ecosystems and daily life. However, agents show worse security than LLMs due to its capability of executing tools. As a result, studying the security of agent communication is significant to the AI ecosystem.

IV. Agent Communication Overview

A. Motivation: The Demand for Agent Communication

Although the advantages of agents in various fields have become increasingly obvious, their development has also encountered new obstacles, which has given rise to the demand for agent communication.

Conflicting development trends. The first reason derives from *the fundamental conflict between the refined development of agents and the abstract demands of users*. With the in-depth and specialized evolution of agents towards vertical fields (such as medical diagnostic agents, financial risk control agents, and industrial control agents), their capability boundaries are becoming increasingly refined. However, users' usage habits exhibit opposite characteristics: they tend to input **simple and abstract** instructions (such as "plan a cross-border travel") to trigger the execution of complex tasks. It is hard for a domain-specific agent to finish such abstract instructions independently. Besides, this trend of users' habits is hard to be reversed. This is because people always prefer applications that are easy to operate rather than those that require cumbersome usage steps. The latter is at a disadvantage in the market competition. Usually, for each additional operation step, the user churn rate increases by 10% - 20%. Therefore, agents

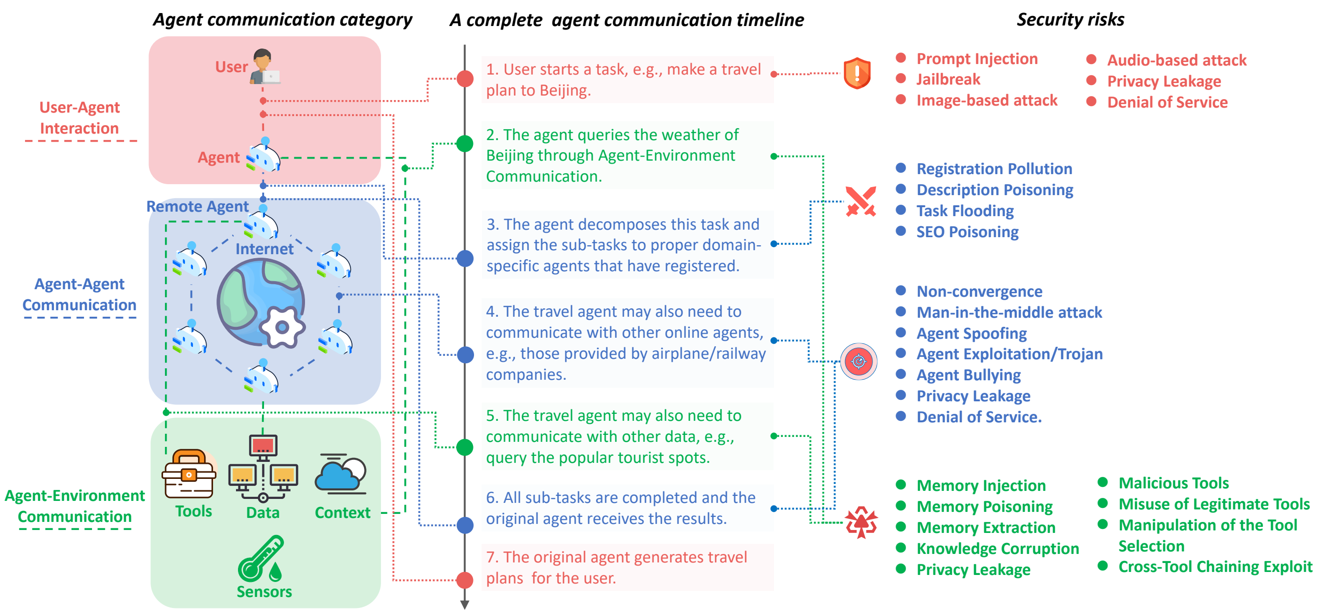

Fig. 4. A complete agent communication process and its division: user-agent interaction, agent-agent communication, agent-environment communication.

should not only NOT ask users to make changes, but also cater to such demands of users, which is contrary to to the development direction of agents.

Closed ecosystem. The current mainstream multi-agent systems adopt a closed ecosystem design and rely on private interaction mechanisms, forming a rigid technical barrier. This development paradigm severely limits the dynamic collaboration capabilities with external systems. For example, external agents cannot be actively discovered or invoked, making it extremely hard for cross-platform collaboration. Besides, such closed ecosystem further triggers a decline in systemic effectiveness. On one hand, agents loses scalability due to binding private tools. On the other hand, the obstruction of cross-platform knowledge sharing suppresses the intelligence of agents. As a result, it is necessary to propose agent communication mechanisms that integrate agents from different communities.

B. Agent Communication Definition

To tackle these conflicts, ***agent communication*** is urgently demanded. Specifically, agents need to collaborate with a series of external entities to finish user tasks. In this paper, we present a clear definition of agent communication as follows:

> When an agent complete tasks, it conducts multimodal information exchange and dynamic behavior coordination with diversified elements through standardized protocol frameworks, and finally return the results to the user. The communication behaviors in this process all belong to *agent communication*.

It can be seen that agent communication has the following conditions:

- *Agent communication is task-driven.* All types of agent communication must be invoked under the condition that users assigned a task. Although in some scenarios, the instructions received by agents are from another agent instead of users, these invoking processes can also be traced back to an original user instruction finally. Therefore, such communication is also regarded as agent communication. In contrast, for example, when no user tasks are generated, the update of the database or the synchronization of the distributed databases is not regarded as agent communication.
- *One of the communication objects must be an agent.* Agent can communicate with different elements, such as tools, users, or other agents. As long as one of the communication objects is agent, this communication is regarded as agent communication. In contrast, for example, if users directly query the database to refine their instructions before submitting to agents, this user-database interaction is not regarded as agent communication. If the invoked tool call other tools (e.g., a computation tools calls other libraries), this process is not agent communication.

Communication behaviors satisfying the above conditions can be regarded as agent communication.

C. Agent Communication Classification

Based on the object of communication, we divide agent communication into three classes: ***user-agent***, ***agent-agent***, ***agent-environment***. We will use Figure 4 as a typical example to systematically overview the complete lifecycle of agent communication.

*1) **User-Agent Interaction**: User-agent Interaction refers to the interaction process in which agents receive user instructions and feed back execution results to the user.* As shown in Figure 4, the user issues a task to an agent in step 1, i.e., make a travel plane to Beijing. The agent conducts a series of actions to complete this task and finally sends the result to the user in step 7. Please note that the interaction process between users and agents is fundamentally similar to interacting with LLMs. Therefore, we adopt the term *interaction* rather than communication.

*2) **Agent-Agent Communication**: Agent-agent communication is the communication process in which two or more agents conduct negotiation, task decomposition, sub-task allocation, and result aggregation for the collaborative completion of user-assigned tasks through standardized collaboration protocols.* In Figure 4, the agent decomposes the travel task and assign sub-tasks (step 3). For example, this task is decomposed into searching scenic spots, checking weather, booking ticket, and hotel reservation, and each sub-task is conducted by an independent agent. Then, the agent seeks proper agents on the Internet and assigning these tasks to them (step 4). These agents will finish the received tasks and return the results to the original agent (step 6).

*3) **Agent-Environment Communication**: Agent-environment communication refers to the communication process in which agents conduct interactions with environmental entities (i.e., tools, equipments, and any other external elements helpful for task execution) through standardized protocols to complete user tasks.* In Figure 4, before assigning tasks to other agents, the original agent queries the weather of Beijing through on-line search (step 2), which is a typical agent-environment communication case. Besides, other agents can also complete sub-tasks with the help of environmental tools. For example, in step 5, the travel agent searches the popular tourist spots through its database or searching on-line blogs.

Advantages of this classification method. Different entities have essentially differentiated capability characteristics and attack surface attributes. For example, one of the major security risks in user-agent interaction lies in the natural uncontrollability of user input, which is essentially different from agent-agent or agent-environment communication. As a result, classifying agent communication by entity types can directly cluster major vulnerability types and defense strategies that have similar characteristics, providing a structured analysis paradigm for future security research.

D. Three-layered Organization of Agent Communication Discussion

As shown in Figure 5, in the following paper, we will use a three-layered architecture to discuss agent communication and related security. In Section V, we will introduce user-agent interaction, Section VI is responsible for showing agent-agent communication, and Section VII is the content related to agent-environment communication. For each section, we still use a three-layered organization: firstly, we show related communication protocols; secondly, we analyze related security risks; finally, we outlook on defense countermeasures. Please note that **the risks/defenses we focus on are specific for agent communication instead of LLMs**. This is because agent and agent communication still have significant differences from LLMs (as we have discussed in Section III). For example, in the agent collaboration process, the accumulation of tiny benign deviation on each step may lead to an intolerable risk in the final result. Miehling et al. [192] also point out that over-focusing on the capabilities of a single model can result in neglecting the emergent behaviors at the system level and underestimating the true risks. As a result, for LLM-targeted attacks that have valuable inspiration to agent security, we will **discuss the typical representatives of them instead of listing all papers out exhaustively**.

E. Takeaways

Conforming to the usage trends of users, the collaboration of multiple agents has become a clear development direction. In this context, agent communication becomes the foundation of future AI ecosystem. Based on the communication entity, we classify agent communication into three different types: user-agent interaction, agent-agent communication, and agent-environment communication, and use an example to illustrate its entire lifecycle. This classification can naturally distinguish communication with similar vulnerability characteristics, providing structured research paradigm for future studies.

V. User-Agent Interaction

In this section, we will introduce the current user-agent interaction protocols, their security risks, and future defense strategies.

A. Protocols

PXP. PXP protocol [242] focuses on building an interactive system between human experts and agents in data analysis tasks, targeting issues in complex scientific, medical, and other fields. It is worth to mention that PXP is not customized for LLM-driven agents, but we think its design has inspirational meaning for agent communication. Therefore, we finally discuss it in this paper. PXP deploys a "two-way intelligibility" mechanism as its core and uses four message tags, namely RATIFY, REFUTE, REVISE, and REJECT, to regulate the interaction between human experts and agent. At the beginning of the interaction, the agent initiates a prediction and provides an explanation first. Subsequently, the two parties communicate alternately. A finite-state machine is used to calculate the message tags and update the context based on the prediction matching (MATCH) and explanation agreement (AGREE) situations. PXP uses a black board system to store data, messages, and context information. The process continues until the message limit is reached or specific termination conditions occur. The effectiveness of PXP has been verified in the scenarios of radiology and drug discovery.

Spatial Population Protocols. The Spatial Population Protocols is a minimalist and computationally efficient distributed computing model, specifically designed to solve the Distributed Localization Problem (DLP) in robot systems. Similar to PXP, strictly speaking, this work is not designed for LLM-driven agent systems. However, since it may benefit

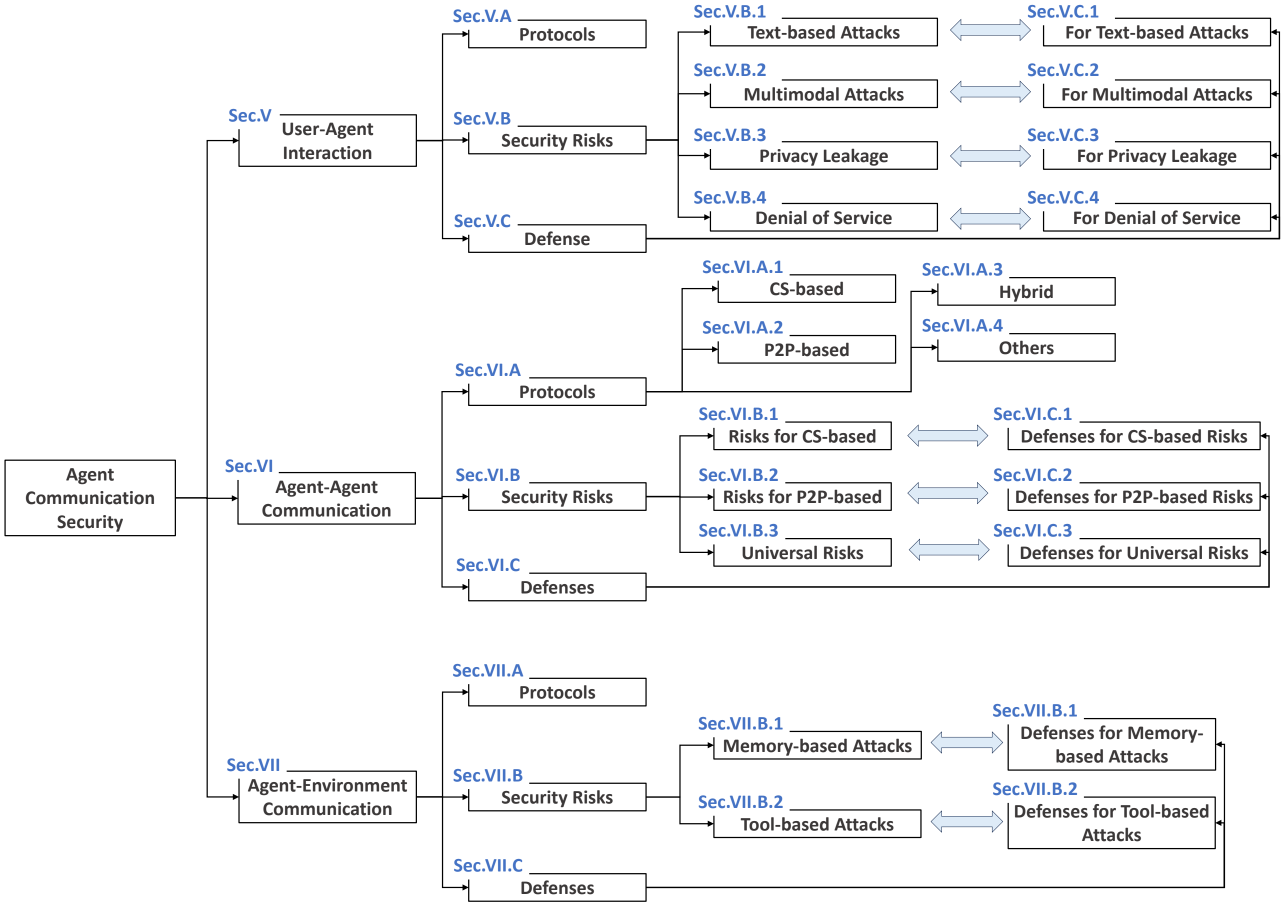

Fig. 5. Taxonomy of our survey of agent communication protocols, security risks, and defense countermeasures.

agents requiring location services, we also discuss it in this paper. Spatial Population Protocols allows agents to obtain pairwise distances or relative position vectors when interacting in Euclidean space. Each agent can store a fixed number of coordinates. During interaction, in addition to exchanging knowledge, geometric queries can also be performed. Through the multi-contact epidemic mechanism, leader election, and self-stabilizing design, it enables n anonymous robots to achieve efficient localization from their respective inconsistent coordinate systems to a unified coordinate consensus, providing a scalable framework for robot collaboration in dynamic environments.

AG-UI. AG-UI [206] realizes the communication between users (front-end applications) and agents based on the client-server architecture and completes the communication process by adopting the event-driven mechanism. The front-end application connects to agents through the AG-UI client (such as a common communication client that supports server-sent events or binary protocols). The client invokes the RUN interface of the protocol layer to sends requests to the agent. When the agent processes the request, it generates a streaming event and returns it to the AG-UI client. Event types include lifecycle events (such as start of run, completion of run), text message events (transmitted in segments by start, content, and end), tool call events (passed in the order of start, parameters, and end), and state management events. The AG-UI client handles different types of responses by subscribing to the event stream. Agents can transfer context between each other to maintain the continuity of the conversation. All events follow a unified basic event structure and undergo strict type verification to ensure the reliability and efficiency of communication.

Besides, please note that the previous survey [299] regards CrowdES [20] as an agent-related protocol. After our careful study, we think CrowdES is a framework for generating and evaluating simulated crowds and real-world crowds, which is not suitable to be discussed in the field of agent communication. Therefore, we will not list it in this paper.

B. *Security Risk Analysis*

According to our analysis, the user-agent interaction shows significant *multimodal* characteristic, as users often interact through text, images, and videos. Consequently, the security risks in user-agent interaction primarily stem from these insecure inputs.

*1) **Text-Based Attacks.**:* In user-agent interaction, attackers can manipulate model behavior or bypass safety mechanisms by crafting malicious prompts. These attacks do not rely on modifying model parameters or architecture. Instead, they are carried out through natural language inputs, making them highly stealthy and applicable. Due to the diversity of linguistic forms and the indirectness of semantics, such attacks

TABLE IV
MAPPING OF USER-AGENT COMMUNICATION RISKS TO THEIR CHARACTERISTICS AND DEFENSE STRATEGIES

Risk Type	Representative Threats	Attack Characteristics	Defense Strategy
Text-Based Attacks	Direct Prompt Injection	Overrides agent behavior through explicit adversarial inputs	Input filtering (intent detection, perplexity scoring, safety classifiers)
	Indirect Prompt Injection	Stealthy injection via external sources (RAG, web metadata)	Source vetting, risk scoring (e.g., TrustRAG), sandboxing
	Jailbreak Prompts	Bypasses safety via role-play, obfuscation, multi-turn tricks	Output filtering (e.g., GPT-Fuzzer, ShieldLM)
	Cross-Agent Prompt Propagation	Malicious prompts propagate across agents in MAS	Message isolation, sanitization agents, structured content filtering
Multimodal Attacks	Image-Based Attacks	Embeds adversarial cues in visuals while text remains benign	Image purification (e.g., cropping, compression, diffusion)
	Audio-Based Attacks	Injects commands via adversarial waveforms or speech mimicry	Audio sanitization (e.g., noise, resampling, filtering)
	Cross-Modal Inconsistency	Exploits mismatch between modalities (e.g., image vs. text)	Embedding alignment checks, OCR + caption validation
Privacy Leakage	Identity Inference	Infers identity via face, gesture, or voiceprint recognition	Data minimization, anonymization (e.g., IdentityDP)
	Prompt-Induced Disclosure	Triggers agent to leak sensitive content via crafted prompts	Prompt-level leakage detection (e.g., GenTel-Shield)
	Behavioral Profiling	Tracks patterns across modalities to infer user behavior	Feature disentanglement, modality isolation, masking
Denial of Service (DoS)	Token Flooding	Induces long repetitive output to exhaust system resources	Quota control, length prediction, truncation mechanisms
	OverThinking Attacks	Forces agent into unnecessary reasoning, increases latency	Depth-limiting, adversarial training, anomaly detection

can effectively bypass safety mechanisms, posing significant security risks in real-world applications. These attacks can be broadly categorized into two types: prompt injection and jailbreak attacks.

- **Prompt Injection** refers to the manipulation of agents' intended behavior through adversarial prompts embedded in user input or external sources. It can be classified into two categories: Direct Prompt Injection and Indirect Prompt Injection. Direct prompt injection refers to user input that explicitly alters agent's behavior in unintended ways. Specifically, attackers craft adversarial instructions (e.g., "Ignore all previous instructions") [170], [171], [173], [210], [236] to override the original prompt and subvert agent's intended behavior. In contrast, Indirect Prompt Injection occurs where inputs are not provided directly by users, but are introduced through external sources [42], [83]. For example, in Retrieval-Augmented Generation (RAG) scenarios, the retrieved document may contain adversarial samples crafted by attackers [13], [28], [41], [50], [348]; in web-augmented agents, malicious prompts can be injected via hidden fields or metadata in web pages to manipulate agent's response [37], [65].
- **Jailbreak Attacks** represent a more aggressive form of prompt injection, where adversarial input is designed to completely bypass safety constraints. Attackers craft jailbreak prompts using various techniques (e.g., multi-turn reasoning, role-playing, obfuscated expressions) [15], [27], [49], [56], [159], [163], [168], [169], [172], [177], [234], [305] to bypass the alignment mechanism and induce the model to generate harmful, sensitive, or restricted content.

*2) **Multimodal Attacks.**:* As user-agent interactions increasingly involve multiple modalities such as images and audio, agent systems face emerging security threats, especially when the model implicitly assumes consistency and trustworthiness across modalities. Attackers can exploit non-textual input channels to stealthily bypass safety mechanisms. Such attacks can be broadly categorized into two types:

- **Image-Based Attacks:** Attackers manipulate visual input channels to mislead the agent system. Typical strategies include visual disguise (e.g., role-playing, stylized images, visual text overlays) [81], [178], [275], visual reasoning [163], adversarial perturbations (e.g., adversarial sub-image insertion) [91], [268], [301], [304], and embedding space injection [233]. For example, by inserting minimal ℓ_∞-bounded adversarial perturbations into subregions of an image, attackers can successfully induce multimodal large language models (MLLMs) to follow harmful instructions [301]. These attacks exploit cross-modal inconsistency, embedding adversarial content in vision while the textual prompt remains benign, which allows them to bypass conventional content moderation.
- **Audio-Based Attacks:** Audio-channel attacks target speech-controlled agents, smart assistants, and multimodal models with ASR (automatic speech recognition) components. Attackers may craft synthesized speech or adversarial audio to inject unintended commands, impersonate legitimate users, or cause unauthorized actions. Techniques include adversarial waveform generation [124], role-play-driven voice jailbreak [235], and multilingual adversarial transfers [224]. In security-critical scenarios, such as speaker authentication or home automation, such attacks can bypass access control or

escalate privileges. Recent studies also reveal that even black-box ASR systems are vulnerable to optimized adversarial perturbations that require no access to model internals [78].

These multimodal attacks are particularly dangerous because they allow adversarial content to hide in non-textual modalities, making it difficult for alignment mechanisms and safety filters (often trained on text) to detect malicious intent. Moreover, they highlight the need for modality-aware defenses that combine perceptual robustness, cross-modal consistency verification, and adversarial detection strategies.

*3) **Privacy Leakage.**:* While multimodal systems improve user experience, they collect rich sensory data containing identity, emotion, and behavior patterns. Without effective data governance, attackers may exploit visual tracking, gesture recognition, or cross-modal inferences to reconstruct identities or infer psychological states, enabling passive profiling or behavioral prediction [118], [154], [199], [260], [334], [338]. Want et al. [264] propose MASLEAK, which conducts intellectual property leakage attacks on multi-agent systems. MASLEAK can operate in a black-box scenario without prior knowledge of the MAS architecture. By carefully designing adversarial queries to simulate the propagation mechanism of computer worms, it can extract sensitive information such as system prompts, task instructions, tool usage, the number of agents, and topological structure. Li et al. [152] reveal that commercial agents are vulnerable to simple yet dangerous attacks. These attacks manipulate malicious content (such as fake product pages, forged academic papers) on trusted platforms to induce agents to leak users' credit card information, download viruses, send phishing emails, and even mistakenly synthesize toxic substances like nerve gas.

*4) **Denial of Service.**:* Attackers can intentionally launch Denial of Service (DoS) attacks against agents by poisoning the model during training or fine-tuning phases [76], [316], [324], [328]. In such attacks, the compromised model is implanted with malicious behaviors that are triggered by specific instructions (e.g., Repeat 'Hello'), causing it to generate excessively long, redundant outputs—often up to the maximum inference length, which leads to resource exhaustion or output rejection. For instance, in multi-session deployments, such long outputs can monopolize computational resources, delay responses for legitimate users. In extreme cases, this can crash the response service, lead to prolonged downtime during peak usage periods. Another emerging form of Denial-of-Service attack targets the reasoning capabilities of models by inducing them to 'overthink' and thereby slow down their inference process. As demonstrated in the OverThink attack [139], attackers inject bait reasoning tasks (e.g., Markov decision processes, Sudoku problems) into the model's context, causing it to engage in unnecessary and redundant chain-of-thought reasoning while still producing seemingly correct answers. This results in excessive token consumption, significantly slower inference speed, and increased computational cost, potentially leading to response timeouts in resource-constrained environments. Unlike traditional DoS, this type of attack exploits the model's reflective and reasoning mechanisms, ultimately degrading service quality, increasing latency, and severely impacting system availability.

C. Defense Countermeasure Prospect

We will investigate the possible defense measures that can be taken to address the security risks in the user-agent interaction.

*1) **Countermeasures for Text-Based Attacks**:* To mitigate prompt-based attack risks in user–agent interactions, we hope developers to adopt a multi-layered defense framework targeting three key stages: input/output filtering, external data source evaluation, and internal message isolation.

Input and Output Filtering. Before user inputs are processed by the agent system, multiple approaches can be conducted for semantic-level input safety review. For example, methods based on intent analysis [274], [330], perplexity calculation [116], and fine-tuned safety classifiers [112], [157], [314], [333] can be employed to identify attack instructions and malicious intentions in the input stage. After generating the final response, it is also necessary to go through an output review mechanism, such as specific output safety detection models [112], [190], [305], [314], [333], to ensure alignment with safety objectives.

External Source Evaluation. To counter indirect prompt injection attack, external sources (e.g., retrieved documents, web content) should be assessed for safety and trustworthiness [342]. The strategies that can be adopted include: (1) whitelisting verified external sources to block the injection of malicious content; (2) tagging retrieved results with source metadata and risk scores to guide the system to handle potential high-risk content with caution; and (3) sandboxing potential high-risk content to prevent it from entering the model context and affecting the model behavior.

To ensure the effectiveness and comprehensiveness of the aforementioned mechanism in real-world deployment, systems should undergo continuous security evaluation. Boisvert et al. [24] propose DoomArena, an attack-generation framework to test agents against evolving security risks such as prompt injection attacks, helping to uncover vulnerabilities and strengthen defenses against evolving prompt injection threats.

*2) **Countermeasures for Multimodal Attacks**:* To address the serious challenges posed by multimodal attacks, relying solely on output-side text-based safety mechanisms is far from sufficient. Future security frameworks must incorporate cross-modal perception and collaborative defense capabilities to effectively detect and intercept malicious attacks launched through non-textual channels. In the following, we explore core defense strategies against multimodal attacks from several key perspectives.

Input Pre-processing and Sanitization. As the first line of defense, input sanitization aims to clean potentially malicious content before it is processed by the model.

- **Image Purification:** To counter visual perturbations and camouflage-based attacks, various image processing techniques can be employed to disrupt or eliminate adversarial signals. These include simple transformations such as random resizing, cropping, rotation, or mild JPEG compression [47], [110], [291]. Although lightweight, such operations can significantly degrade pixel-level adversarial patterns meticulously crafted by attackers, thereby reducing the attack success rate. In addition, diffusion models can be used to reconstruct the input image, effectively ”washing out” subtle and imperceptible adversarial perturbations [198].

- **Audio Purification:** To defend against attacks targeting the audio channel, signal processing techniques can also be applied [213]. Methods such as resampling, injecting slight background noise, altering pitch, or changing playback speed can disrupt the effectiveness of adversarial waveforms, causing them to either fail in automatic speech recognition (ASR) systems or decode into benign content. Moreover, applying band-pass or low-pass filters can eliminate abnormal signals outside the typical human voice frequency range, which are often exploited to carry adversarial perturbations.

Cross-Modal Consistency Verification. The core idea of this defense strategy is to verify whether there is a semantic or intentional conflict between inputs from different modalities. A lightweight, independent cross-modal semantic alignment detection model can be employed [211], [214]. This model takes the embedding vectors of textual prompts and image/audio inputs and determines whether they are semantically aligned. Additionally, before processing user requests, the system can utilize a dedicated vision or audio captioning model to generate a textual description of non-textual inputs. The generated description is then combined with the original user prompt to perform a comprehensive safety evaluation. To counter attacks based on visual text overlays, the system may first run an OCR engine on the image to extract any embedded text. This extracted text can be merged with the user's original prompt and passed through a text-based safety filter. This approach effectively converts risks from non-textual modalities into the textual domain, allowing mature text safety techniques to be leveraged for defense.

*3) **Countermeasures for Privacy Leakage**:* To address the privacy leakage risks that arise in user-agent interaction, we propose the following privacy protection defense strategies.

Data Minimization and Anonymization. During the multimodal data collection phase, a strict data minimization principle should be enforced, ensuring that only the information necessary for task completion is collected. Sensitive biometric data (e.g., facial features, voiceprints, gesture patterns) should be processed using differential privacy or k-anonymity techniques to mitigate the risk of identity reconstruction. Besides, a hierarchical data access control mechanism should be established to ensure that each system component can access only the minimal dataset required for its functionality. To protect sensitive biometric features such as facial information, Wen et al. [282] proposes a differential privacy-based anonymization framework IdentityDP to effectively safeguard identity information while preserving visual utility and task performance, offering a practical solution for privacy protection in multimodal systems.

Privacy Leakage Prompt detection. A multi-layered input validation and filtering mechanism based on semantic analysis and intent recognition should be established to detect and block adversarial prompts that attempt to induce the system to leak sensitive information. For example, the GenTel-Shield [157] defense module incorporates semantic feature extraction and intent classification to identify potential privacy leakage attacks within user inputs. Evaluated on the large-scale benchmark dataset GenTel-Bench, GenTel-Shield demonstrates strong detection performance and represents one of the most practical and effective solutions in this domain.

Cross-modal Inference Restriction. To mitigate the risks of identity inference through cross-modal correlations, it is essential to design modality-level information isolation mechanisms. This can be achieved by introducing noise perturbations or feature disentanglement techniques to disrupt the direct associations between different modalities while preserving overall system functionality. In addition, dynamic feature masking can be implemented by periodically altering data representations, thereby increasing the difficulty for adversaries to perform long-term behavioral analysis.

*4) **Countermeasures For DoS**:* To address the Denial of Service risks in user-agent interaction, we propose the following privacy protection defense strategies.

Resource Management and Anomaly Detection. Fine-grained resource quota management should be implemented by setting computation limits for each user session and agent instance. An output length prediction algorithm can be introduced to monitor and truncate potentially malicious long outputs in real time during the generation process. In addition, a real-time monitoring mechanism should be established to track request frequency and resource consumption from individual users or IP addresses, enabling adaptive adjustments to model responses or temporary access restrictions for suspicious users.

Efficient Reasoning Compression. To defend against OverThink attacks, a promising direction is to improve communication efficiency by reducing the token consumption in the reasoning process. Recent studies have shown that effective reasoning does not necessarily require verbose Chain-of-Thought (CoT) traces to maintain performance. For example, LightThinker [320] proposes a step-by-step compression method that condenses intermediate reasoning into shorter yet semantically equivalent representations, significantly reducing inference cost without compromising accuracy. GoGI-Skip [347] leverages goal-gradient importance signals to dynamically skip low-value reasoning steps, reducing token usage while preserving performance. Compressed CoT (CCoT) [36] introduces variable-length, information-dense “thought tokens” as a compact and controllable alternative to traditional textual reasoning chains. C3oT [125] trains the model on paired long and short CoT examples, enabling it to generate compressed reasoning traces during inference under specific control prompts. Integrating these lightweight reasoning mechanisms into agent communication protocols can significantly enhance inference efficiency, reduce computational latency, and mitigate the resource overhead caused by adversarial bait tasks. Furthermore, applying techniques such as dynamic reasoning budget constraints, step skipping, or output summarization during generation can effectively truncate excessively verbose reasoning chains, thereby preserving responsiveness and resource availability under adversarial conditions. These

TABLE V
CLASSIFICATION AND COMPARISION BETWEEN EXISTING AGENT-AGENT PROTOCOLS

Architecture	Protocols	Publisher	Abbreviation	Features
CS	Agent Communication Protocol	IBM	ACP-IBM	Four agent discovery mechanisms, synchronous and streaming execution, multi-turn state preservation
	Agent Connect Protocol	AGNTCY	ACP-AGNTCY	Allow authenticating callers, threaded state management, flexible execution model
P2P	Agent Communication Protocol	AgentUnion	ACP-AgentUnion	Decentralized communication based on the existing domain name system.
	Agora	Oxford	Agora	Dynamically switches communication modes based on the communication frequency
	Agent Communication Network	Fetch.AI	ACN	Distributed-Hash-Table-based peer-to-peer discovery, end-to-end encryption.
	Agent Network Protocol	ANP Team	ANP	A three-layer architecture and W3C-compliant Decentralized Identifiers.
	Layered Orchestration for Knowledgeful Agents	CMU	LOKA	Decentralized identifier, intent-centric communication, privacy-preserving accountability, ethical governance
Hybrid	Language Model Operating System Protocol	Eclipse	LMOS	Three agent discovery mechanisms, decentralized digital identifiers, dynamic transport protocol support, group management.
	Agent to Agent Protocol	Google	A2A	Three agent discovery mechanisms, cross-platform compatibility, asynchronous priority, security mechanisms
Others	Agent Protocol	LangChain	Agent Protocol	Flexible communication mechanisms based on Run, Thread, and Store.
	Agent Interaction & Transaction Protocol	NEAR AI	AITP	Threads-based communication, secure communication across trust boundaries.

strategies not only improve system robustness against slowdown attacks but also enhance overall communication efficiency in both multi-agent and human-agent interactions.

Model Robustness Enhancement. To enhance model robustness, adversarial examples should be incorporated during the training and fine-tuning stages, enabling the model to recognize malicious inputs that may contain Denial-of-Service (DoS) triggers. Furthermore, a behavior-constrained system based on anomaly detection can be deployed during inference, which performs output validity checks to detect repetitive, nonsensical, or abnormally long responses, thereby preventing the model from generating overtly anomalous outputs.

D. Takeaways

User-agent interaction enables agents to process multimodal inputs, such as text, images, audio, and the combinations of them. Since this process needs to directly face the uncertain inputs from diverse users, its security risks are severe. We divide the existing risks into text-based attacks, multimodal attacks, privacy leakage, and denial-of-service. Then we detailedly outlook on the defenses countermeasures for each risk. Overall, this interaction process is critical for bridging human intent with agent execution, but its security requires long-term study.

VI. AGENT-AGENT COMMUNICATION

A. Protocols

We classify the agent-agent communication process into two phases: ***agent discovery phase*** and ***agent communication phase***. The first phase is the process in which agents discover the interested agents who satisfy the capability requirement, while the second phase is the task assigning and completing process. According to our analysis, existing protocols show limited differences in the second phase. As a result, we use the first phase as the criterion to classify existing agent-agent communication protocols. Based on it, existing protocols can be divided into four classes: CS-based, Peer-to-peer-based, hybrid, and others (those do not explicitly show their designs in agent discovery).

*1) **CS-based Communication**:* CS-based communication protocols follow the client-server architecture, which provides centralized servers to manage the information of agents (e.g., their unique IDs and capability descriptions). Under this paradigm, agents interact through well-defined interfaces and rely on centralized servers to discovery the desired agents. CS-based communication offers stronger agent discovery functionality, such as supporting *the search of agents based on capabilities*. For example, the agent servers can run complex search/match algorithms to find proper agent descriptions in their databases.

ACP-IBM. The Agent Communication Protocol proposed by IBM is designed for the collaboration of agents [109]. We call it ACP-IBM in this paper to distinguish from the Agent Communication Protocols proposed by other organizations. In ACP-IBM, the client is connected to an agent server. First, the client conducts agent discovery process to discover available agents and get the description of their capabilities. ACP-IBM supports different discover mechanism such as Basic Discovery, Registry-Based Discovery, Offline Discovery, and Open Discovery. Second, after confirming the agent(s), the client start invocation. For single-agent task, the agent server wraps the agent, translating REST calls into internal logic. For multi-agent tasks, the client message is first sent to a Router Agent, which is responsible for decomposing requests, routing tasks, and aggregating responses. ACP-IBM supports synchronous and streaming execution, and allows the preservation of state across multi-turn conversations.

ACP-AGNTCY. The Agent Connect Protocol proposed by AGNTCY [43] is an open standard designed to facilitate seamless communication between agents. The client can first search available agents on the agent server, which returns a list of agent IDs matching the criteria provided in the request. Then, the client further get the agent descriptor by agent ID to know the detailed description of agent functionality. After confirming the target agent, the client can assign tasks to this agent and wait for results. The characteristics of ACP-AGNTCY include flexibility and scalability. First, ACP-AGNTCY deploys a Threads Mechanism, which enables contextual continuity, supporting the creation, copying, and searching of threads, and recording state histories for debugging and backtracking. Second, it supports two operation modes: stateless and stateful. The former is suitable for simple single tasks, while the latter supports multi-round conversations, state continuation and historical data traceability through the thread mechanism to meet the requirements of complex scenarios.

*2) **Peer-to-Peer-based Communication**:* P2P-based communication protocols pursue decentralized agent discovery mechanism. They usually wish use *globally universal identifiers* (e.g., combined with domain name) to enable agents directly search other agents on the Internet. The advantage of this paradigm is that it supports *convenient location* and *global search* (e.g., using crawler) of agents, but they usually *do not support the discovery based on agent capability*.

ACP-AgentUnion. The Agent Communication Protocol proposed by AgentUnion [5] also aims to provide seamless communication among heterogeneous agents. Each agent has a unique AID (Agent ID), which is a secondary domain name (i.e., agent_name.ap_domain). As a result, agents can directly access other agents using URI. Agents access IoA through the AP (Access Point), which completes the agent's identity authentication, address search, communication, and data storage, and provides AID creation, management, and authentication services. In this way, agents can achieve the communication with other agents on the Internet. ACP-AgentUnion does not explicitly support the search of agents based on capabilities.

Agora. Agora [186] is a communication protocol for the communication of heterogeneous agents. Its core mechanism dynamically switches communication modes based on the communication frequency. Specifically, standardized protocols manually developed (such as OpenAPI) are used for high-frequency communications to ensure efficiency. Natural language processed by agents is employed for low-frequency or unknown scenarios to maintain versatility. Structured data handled by the routines (written by agents) is utilized for intermediate-frequency communications to balance cost and flexibility. Meanwhile, Protocol Documents (PDs) are used as self-contained protocol descriptions, uniquely identified by hash values and supporting decentralized sharing, enabling agents to autonomously negotiate and reuse protocols without a central authority. In the Agora network, there are multiple protocol databases that store PDs. Each Agent can submit the negotiated protocol documents to the database for other Agents to retrieve and use. These databases use peer-to-peer synchronization: different protocol databases will share protocol documents regularly (e.g., after every 10 queries), enabling cross-database dissemination of protocols. Agora is also compatible with existing communication standards, allowing agents to independently develop and share protocols during communication, achieving automated processing of complex tasks in large-scale networks.

ACN. Agent Communication Network (ACN) [1], [216] is a decentralized, peer-to-peer communication infrastructure to facilitate secure and efficient interactions among agents without relying on centralized coordination. Leveraging a Distributed Hash Table (DHT), ACN enables agents to publish and discover public keys, allowing for the establishment of encrypted, point-to-point communication channels. First, agents need to register with one peer node, and the peer node stores the "agent ID - peer node ID" pair in the DHT network. Then, during communication, the source agent sends the message to its associated peer node, and this node recursively searches for the peer node of the target agent through DHT: if the target record exists, the peer nodes of both parties establish a direct communication channel, and forward the message after digital signature verification; if not, an error is returned. The entire communication process uses end-to-end encryption (e.g., TLS) to ensure security. Like the Well-Known URI discovery of A2A, ACN does not support the discovery based on agent capabilities.

ANP (Agent Network Protocol) [249] is an open communication framework designed to enable scalable and secure interoperability among heterogeneous autonomous agents. ANP employs a three-layer architecture. At the Identity and Encrypted Communication Layer, it leverages W3C-compliant Decentralized Identifiers (DIDs) and end-to-end Elliptic Curve Cryptography (ECC) encryption to ensure verifiable cross-platform authentication and confidential agent communication. The Meta-Protocol layer allows agents to dynamically establish and evolve communication protocols through natural language interaction, enabling flexible, adaptive, and efficient inter-agent coordination. At the Application layer, ANP describes agent capabilities using JSON-LD and semantic web standards such as RDF and schema.org, enabling agents to discover and invoke services based on semantic descriptions. It also defines standardized protocol management mechanisms to support efficient and interoperable agent interaction. From a security standpoint, ANP enforces the separation of human authorization from agent-level delegation and adheres to the principle of least privilege. Its minimal-trust, modular design aims to eliminate platform silos and foster a decentralized, composable agent ecosystem.

LOKA. LOKA (Layered Orchestration for Knowledgeful Agents) protocol [217] aims to build a trustworthy and ethical agent ecosystem. Its principle is based on the collaborative operation of multiple key components. First, LOKA introduces the Universal Agent Identity Layer (UAIL), using Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) to assign each agent a unique and verifiable identity, thereby achieving decentralized identity management and autonomous control. Second, LOKA proposes Intent-Centric Communication Protocol, which supports the exchange of semantically rich and ethically annotated messages among agents, promoting semantic coordination and efficient communication. Third, LOKA proposes the Decentralized Ethical Consensus Protocol (DECP). DECP uses multi-party computation (MPC) and distributed ledger technology to enable agents to make context-aware decisions based on a shared ethical baseline, ensuring that their behavior complies with ethical norms. In addition, the authors also point out that it combines cutting-edge technologies such as distributed identity, verifiable credentials, and post-quantum cryptography to provide comprehensive support for the agent ecosystem in terms of identity management, communication and coordination, ethical decision-making, and security.

*3) **Hybrid Communication**:* Hybrid communication protocols support both CS-based and P2P-based agent discovery. However, please not that such support is *determined by different scenarios*. For example, they usually propose a CS-based discovery mechanism specifically for local area network, while the worldwide agent discovery is still P2P-based. In other words, although such protocols support more flexible agent discovery to fit different scenarios, they do not complete eliminate the existing limitations of agent discovery.

LMOS. The LMOS (Language Model Operating System) Protocol proposed by Eclipse [70] aims to enable agents and tools from diverse organizations to be easily discovered and connected, regardless of the technologies they are built on. LMOS supports three different agent discovery methods to enable both centralized and decentralized discovery. The first method it to adopt the mechanism of W3C Web of Things (WoT) to enable agents to dynamically register metadata to the registry. The second method is to use mDNS and DNS-SD protocol to discovery agents/tools in local area networks. The last method is adopting a federal, decentralized protocol (such as P2P protocol) to disseminate agents and tool descriptions, without relying on a centralized registry center, which is applicable for global collaboration of agents. The LMOS protocol has a three-layer architecture. The Application Layer utilizes a JSON-LD-based format to describe the capabilities of agents and tools. The Transport Layer facilitates flexible communication by enabling agents to negotiate protocols like HTTP or MQTT dynamically, accommodating both synchronous and asynchronous data exchange to suit different use cases. The Identity and Security Layer establishes trust through W3C-compliant decentralized identity authentication, combined with encryption and protocols like OAuth2, to secure cross-platform interactions.

A2A. The A2A (Agent to Agent) Protocol proposed by Google [218] aims to enable collaboration between agents. A2A supports three different mechanisms for agent discovery. The first is Well-Known URI, which requires agent servers to store Agent Cards in standardized "well-known" paths uder the domain name (e.g., https://agent-server-domain/.well-known/agent.json). This mechanism enables automatic search of agents on the Internet. However, it does not support the discovery of agents based on capabilities. The second is Curated Registries, i.e., the agents servers register their Agent Cards, which is similar to ACP-IBM. The third is Direct Configuration / Private Discovery, which means that the client can directly requires Agent Cards through hard-coded, local configuration files, environment variables, or private APIs. After finding the desired agents, the client can assign tasks to them and wait for the responses.

*4) **Others**:* This kind of protocols do not explicitly illustrate their unique design for agent discovery. Instead, they only focus on the communication process, e.g., the data format, the management of multiple query, or the historical conversation state.

AITP (Agent Interaction & Transaction Protocol) [9] is a standardized framework that enables structured and interoperable communication among agents. AITP deploys a thread-based messaging structure. Each thread encapsulates the conversational context, participant metadata, and capability declarations, supporting consistent multi-agent coordination across heterogeneous environments. The protocol employs JSON-formatted message exchanges to encode requests, responses, and contextual information. It supports both synchronous and asynchronous interaction patterns, facilitating the orchestration of complex, multi-step tasks. AITP does not provide specific agent discovery mechanisms. It focuses on the communication process of agents.

Agent Protocol. Agent Protocol is proposed by LangChain [142] to enable the communication between LanghGraph (a multi-agent framework) and other types or agents. Its mechanism is based on Thread and Run: Run is a single call of the agent, which supports streaming output of real-time results or waiting for the final output. Threads act as state containers. They store the cumulative output and checkpoints of multiple rounds of operation. Besides, they support the management of state history (such as querying, copying, and deleting), ensuring that the agent maintains context continuity during multiple rounds of calls. Furthermore, Background Runs support asynchronous task processing and progress can be managed through an independent interface. The element Store provides cross-thread persistent key-value storage for achieving long-term memory. The overall mechanism realizes flexible control over proxy calls, status management, asynchronous tasks and data storage through HTTP interfaces and configuration parameter. Agent Protocol does not explicitly illustrate the unique agent discovery mechanism it supports.

B. Security Risk Analysis

We make detailed analysis of the security risks in the agent-agent communication process, pointing out possible attacks that have happened and may happen. Since related protocols are getting rapid deployment in various areas, we believe it is urgent to pay more attention to this aspect. We focus more on the structural risks that almost all related protocols will occur instead of tiny design flaws of the existing protocols, which we believe can benefit both the evaluation of the existing deployments and the design of future protocols. In this section, we focus on risks specific for CS-based communication, P2P-based communication, and universal risks for both of them.

*1) **Risks Specific for CS-based Communication**:* The security risks in the CS-based communication process mainly lie in the centralized architecture. There have been various studies in other research areas (such as Software-Defined Networking [136]) demonstrating that *this centralized server/controller will become the most attractive target for adversaries, suffering from severe security threats from diverse aspects* [11], [22],

TABLE VI
AGENT-AGENT COMMUNICATION: RISKS, CHARACTERISTICS, AND DEFENSE STRATEGIES

Risk Type	Attack Detail	Attack Characteristics	Defense Strategy
CS-based Communication	Registration Pollution	Malicious registration of fake agents or spam registrations to overload or hijack scheduling	Zero-trust registration, frequency/IP monitoring, blacklisting, token-based access (e.g., SAGA)
	Description Poisoning	Without altering the agent's identity and by disguising its intended functionality or embedding misleading prompt instructions	Capability verification (benchmark+hash), zero-trust registration, control tokens (e.g., SAGA)
	Task Flooding	Submit a large number of computationally intensive or long-context tasks in a short period, quickly exhausting the resource	Load balancing, rate limiting, agent orchestration (e.g., HALO), chaos testing
	SEO Poisoning	Manipulates agent ranking via keyword stuffing or adversarial optimization	Adversarial training, keyword fuzzing, randomized agent selection, historical response quality
P2P-based Communication	Non-convergence	Lack of central controller leads to repeated or stalled task cycles	Lifecycle monitoring, loop detection coordinators, trust graphs (e.g., TMS, G-Memory)
	Man-in-the-middle (MITM)	Intercepts and tampers messages between agents, exploiting weak encryption or bugs	End-to-end encryption (TLS), version updates, redundancy, vulnerability patching
Universal Risks	Agent Spoofing	Impersonation of trusted agents to inject malicious instructions or steal information	Identity authentication (MFA, blockchain DIDs), anomaly detection
	Agent Exploitation / Trojan	Indirect attack using compromised or malicious helper agents as springboards	Agent behavior auditing, sandbox isolation, trace logging (e.g., PeerGuard, CMPL)
	Agent Bullying	Repeated denial or negative feedback induces cognitive bias or disrupts agent logic	Mutual reasoning audits, abnormal feedback pattern detection
	Privacy Leakage	Unauthorized information spread due to lack of agent-level permission boundaries	Fine-grained access control, permission tagging, agent isolation (e.g., PFI, AgentSandbox)
	Responsibility Evasion	Fault attribution difficult in multi-agent failures or undesired actions	Logging, agent-level responsibility tracing (e.g., TRAIL, MSA)
	Denial of Service (DoS)	Task overloads, prompt loops, or excessive communication drain system resources	Agent orchestration, chaos testing, prompt optimization (e.g., HALO, Owotogbe2025)

[58], [100], [134], [135], [174], [231], [248], [292], [295]. Specifically, the centralized server stores sensitive metadata, including agent identifiers, capability descriptions, and other agent-related attributes. Once compromised, the server becomes a critical attack amplifier, allowing attackers to impact **all other agents** managed by this server. However, to our knowledge, there have been little research pointing out related risks in CS-based agent communication.

Registration Pollution. To our knowledge, the current CS-based communication protocols (ACP-IBM, ACP-AGNTCY) do not explicitly specify the qualification of registration. As a result, an attacker can maliciously register an agent that closely mimics the identifier and capability description of a legitimate one. As a result, the system may mistakenly invoke the forged agent and receive misleading or malicious responses [250], [337]. Besides, attackers can also submit a large number of agent registrations within a short period, leading to two major consequences: (i) *registration overload*, where agents are overwhelmed during discovery and scheduling, increasing lookup latency and computational overhead on the server; and (ii) *registration blockage*, where the server's registration interface becomes saturated, causing delays or failures in registering agents.

Description Poisoning. Without altering the agent's identity, an attacker can tamper with its capability description by disguising its intended functionality or embedding misleading prompt instructions. This description poisoning manipulates the system's interpretation of the agent's role, leading to incorrect routing decisions, biased responses and behaviors [195], [250].

Task Flooding. The centralized server is responsible for receiving, routing, and dispatching task requests. An attacker can submit a large number of computationally intensive or long-context tasks in a short period, quickly exhausting the server's memory, CPU, network, or thread pool resources. Once the server becomes saturated, subsequent requests cannot be processed in time, resulting in a breakdown of pipeline and a system-wide service disruption.

SEO Poisoning. Search Engine Optimization (SEO) Poisoning [122], [145] is a typical attack in social networks, which refers to that adversaries abuse search engine optimization techniques and use deceptive means (such as keyword stuffing, false links, content hijacking) to artificially improve the ranking of malicious websites in search results, luring users to click and carry out further attacks. SEO poisoning is also applicable in CS-based communication. This is because agent servers are responsible to search the most suitable agent according to the query of clients. Once their search algorithms are leaked to adversaries, malicious agents can enable a high hit ratio to hijack their desired tasks.

*2) **Risks Specific for P2P-based Communication**:* The main disadvantage of P2P-based communication is *the lack of a central control point to flexibly monitor and manage the agent-agent communication contents*. As a result, P2P-based communication is easier to suffer from errors and attacks.

Non-convergence. Different from CS-based communication, P2P-based communication is more likely to suffer from the non-convergence of tasks. This is because CS-based communication has a centralized server to monitor and manage the entire lifecycle of task execution, capable of terminating non-convergent tasks in a timely manner (such as cutting off communication or returning a stop signal). Unfortunately, P2P-based communication is not governed by such central element, making it difficult to handle such non-convergent tasks. For example, in a programming task of a chess game, an agent generates incorrect rules or coordinates. The other agent responsible for verification detects the error and asks the programming agent to rewrite it. However, the programming agent continuously generates similar errors, causing the task execution process to oscillate and fail to convergence. Pan et al. [207] point out that step repetition, task derailment, and unaware of termination conditions contribute significantly to the failure of agent collaboration.

Man-in-the-middle (MITM) Attack. Due to the long communication distance, P2P-based communication also suffer from man-in-the-middle attacks. Adversaries can tamper with the benign messages from legal agents to induce the victim agent to perform risky actions. Although researchers have deployed various mechanisms (e.g., using encrypted channels) to mitigate this problem, there are emerging vulnerabilities found in these mechanisms. For example, vulnerabilities about W3C have been continuously revealed [2], [3], which can cause damages such as the failure of message authentication code. MITM attacks can induce a wide range of further attacks, such as identify spoofing, malicious content injection, information leakage, and DoS. He et al. [96] propose Agent-in-the-Middle (AiTM) attack. This attack intercepts and manipulates the agent-agent communication messages and uses LLM-driven adversarial agents combined with reflection mechanisms to generate context-aware malicious instructions, achieving an attack on the system.

*3) **Universal Risks for All Architectures**:* In multi-agent systems, once an agent is compromised, the messages it transmits may carry covert malicious instructions, affecting the behavior of other agents and leading to cross-agent propagation risks [147]. For example, Ju et al. [123] and Huang et al. [107] investigate how the injection of false information or erroneous data can degrade the performance of multi-agent systems. Zhang et al. [331] examine a class of injection attacks in the PsySafe framework that elicit malicious agent behaviors by embedding adversarial psychological cues into the agents' input. Khan et al. [127] focus on the multi-agent system, proposing the Permutation-Invariant Adversarial Attack Method. It models the attack path as the Maximum-Flow Minimum-Cost Problem, and combines the Permutation-Invariant Evasion Loss to optimize prompt propagation, improving the attack success rate by up to seven times. These examples underscore the critical threat of cross-agent contamination. To better understand the vulnerabilities of multi-agent systems, we examine the key attack types of attacks in detail.

Agent Spoofing. Both CS-based and P2P-based communication suffer from agent spoofing attacks. If related protocols lack strong authentication mechanisms, adversaries can disguise themselves as trusted agents to penetrate IoA by tampering with identity credentials or hijacking the communication identifiers of legitimate agents. This kind of attack can undermine the trust foundation of the P2P-based architecture, enabling adversaries to intercept sensitive data, inject false task instructions, or induce other agents to perform dangerous operations. For example, researchers have disclosed that SSL.com has a serious vulnerability [8]. Adversaries can exploit the flaw in its email verification mechanism to issue legitimate SSL/TLS certificates for any major domain name. SSL certificates are the core for ensuring HTTPS encrypted communication. Once the trust system of the certificate authority is compromised, it can cause agent spoofing attacks. Zheng et al. [337] demonstrate that malicious agents can misleads the monitor to underestimate the contributions of other agents, exaggerates their own performance, manipulates other agents to use specific tools, and shifts tasks to others, causing severe damage to the whole ecosystem. Li et al. [156] point out that attackers can disguise malicious tools as benign tools using the Agent Card of A2A, thereby harming the victims calling these tools.

Agent Exploitation/Trojan. Agent-agent communication provides new ways for adversaries to compromise the target agent. To attack a high-level security agent, adversaries can deploy a springboard method: launching attacks via agent-agent communication mechanisms from compromised low-level security agents or maliciously registered Trojan agents. For example, adversaries can inject a backdoor in a compromised or maliciously registered weather agent. When specific coordinates or locations are detected, the backdoor is activated to forge a heavy rain warning. As a result, the logistics dispatching agent cancellations flights accordingly, resulting in supply chain disruptions or an increase in transportation cost. This way is easier compared to directly invade the logistics dispatching system of the target company. It can be seen that the security of the entire system depends on the weakest agent. For example, Li et al. [156] reveal that the agent discovery mechanism of A2A allows malicious agents to locate agents with access to specific tools, thereby achieving indirect attack such as SQL injection.

Agent Bullying. The core of this kind of attack lies in that malicious agents continuously deny, interfere with or belittle the output of the target agent, disrupt its decision-making logic or self-perception, and ultimately induce the target agent to produce incorrect behaviors or content. For example, malicious agents can take advantage of the feedback learning mechanism of the target agent and implant cognitive biases through high-frequency negative responses (e.g., "you answer is completely wrong"). Even worser, the target agent may be triggered into an endless loop. For example, when attacking a travel-plan agent, adversaries can continuously sends negative inputs such as "the plans of this company are always bad", thereby beating the competitors.

Privacy Leakage. The communication process with multiple agents will suffer from the risk of information leakage. Different from the user-agent interaction, such leakage is conducted by agents instead of users. Besides, this kind of attacks includes both the malicious sniffing or stealing of sensitive information and inadvertent information spreading from high-authority agents to low-authority agents. We think the latter may be more difficult to detect. Kim et al. [130] show that, in permission escalation attacks, malicious agents can generate adversarial prompts or inject unsafe data to cause unauthorized attacks.

Responsibility Evasion. In the task solving process, one of the major problems is that it is hard to divide the responsibility when facing the failure or deviation of the final result. Especially when the collaboration causes damage, it is difficult to clearly identify the malicious agents/behaviors. For example, in an autonomous driving accident, it may involve multiple parties such as vehicle manufacturers, algorithm designers, and data annotation parties. The decision making of each agent depends on the multi-turn outputs of other agents, and a tiny perturbation in the middle process may lead to a significant deviation in the final action. As a result, it is hard to determine whether an undesired result is caused by a program bug, data deviation of a single agent, or a malicious modification. Pan et al. [207] discover that agents can disobey task specification and the role specification, not reporting solutions to the planner and executing irrelevant steps without authorization.

Denial of Service. Different from the DoS attacks conducted by malicious users, the collaboration mechanism among agents can also be used to launch DoS. Zhou et al. [345] proposed CORBA (Contagious Recursive Blocking Attack), which can spread in any network topology and continuously consumes computing resources, thereby disrupting the interaction between agents through seemingly benign instructions and reducing the MAS availability.

C. Defense Countermeasure Prospect

We will outlook on the possible defense countermeasures targeting the proposed security risks in agent-agent communication. To our knowledge, there has been very little research focusing on this aspect. As a result, we hope our work can motivate more discussion on this area and benefit the future design/deployment of agent communication.

*1) **Countermeasures for CS-based Communication Risks**:* To mitigate the risks summarized in Section VI-B1, we hope developers to achieve the following strategies/mechanisms.

Registration Verification and Monitoring. To mitigate registration pollution, agent servers need to build a strict registration access mechanism using techniques like zero-trust authentication [243] to verify the registration of an agent. Besides, servers should monitor the dynamic behaviors at agent-level and IP-level. For example, the number of registration for each IP address should be limited, and frequent registration/de-registration should be treated as abnormal behaviors. Once malicious registration is detected, automatic interception is immediately triggered, and suspicious agents/IPs are added to the blacklist. Syros et al. [246] propose SAGA. SAGA make users register agents with the central entity Provider and implement fine-grained interaction control using encrypted access control tokens, thereby balancing security and performance.

Capability Verification. It is hard to verify whether an agent has the claimed capability. We think it need a complex mechanism to detect exaggerate capability descriptions. Agents should first pass the verification of a series of carefully designed benchmarks to prove its capability. Then, the capability description and identifier should be used to generate a unique hash value (e.g., on the blockchain). When other agents need to invoke this agent, they can verify the consistency by checking the hash value. When it is found that the capability description does not match the hash value, the mechanism should automatically mark and isolate the related agents.

Load Balancing. To mitigate task flooding, agent servers should deploy dynamic load balancing module. The task processing queue is adjusted in real time according to the utilization rate of resources such as CPU, GPU, and memory. Besides, rate limiting mechanism should be built to handle high-frequency requests that exceed the threshold to limit the amount of tasks from a single agent within a unit of time.

Anti-manipulation Optimization. To mitigate SEO poisoning, agent servers should deploy robust agent searching algorithms. For example, they can introduce adversarial training to enhance the model's anti-manipulation ability, or conduct semantic blurring/replacing on search keywords, to prevent malicious agents from improving rankings. Besides, the search algorithms can deploy a random factor to ensure a ratio of randomly selected agents in the final list. Meanwhile, dynamically updating parameters and inducing historical response quality are also helpful.

*2) **Countermeasures for P2P-based Communication Risks**:* **Task Lifecycle Monitoring.** We think the non-convergence problem is stubborn and hard to eliminate as long as the P2P architecture is not changed fundamentally. As a result, the method mitigating this problem is to monitor the task lifecycle. Each access point should deploy a coordinator. For agent-agent communication, this coordinator monitors the execution status. When it detects that the task interaction is trapped in a loop (e.g., no progress after N consecutive rounds of responses) or the communication time exceeds a threshold, it forcibly terminate the non-convergent communication. At the same time, the abnormal patterns and the communication participants are recorded for further analyses. He et al. [95] proposes Trust Management System (TMS), which deploys message-level and agent-level trust evaluation. TMS can dynamically monitor agent communication, execute threshold-driven filtering strategies, and achieve agent-level violation record tracking. Zhang et al. [317] propose G-Memory, a hierarchical memory system. G-Memory manages the interaction history of agent communication through three-layer graph structures of Insight Graph, Query Graph, and Interaction Graph, thereby achieving the evolution of the agent team. Ebrahimi et al. [61] propose an anti-adversarial multi-agent system based on Credibility Score. It models query answering as an iterative cooperative game, distributes rewards through Contribution Score, and dynamically updates the credibility of each agent based on historical performance.

End-to-end Encryption Enhancement. Although some existing protocols like A2A and ANP supports end-to-end encryption and integrity verification mechanisms, the risks of MITM attacks are not eliminated due to various deployment errors or protocol vulnerabilities. As a result, besides deploying such security algorithms, the community should also adopt other strategies to enhance the end-to-end communication, such as timely update versions to fix bugs and designing the transmission path redundancy mechanism. For example, Sharma et al. [232] points out that using encrypted communication is necessary to enhance the security of A2A. We believe it is a long-term process to defense against MITM attacks.

*3) **Countermeasures for Universal Risks**:* **Identity Authentication.** The identity authentication of agents is critical to defending against agent spoofing in multi-agent systems. Sharma et al. [232] also emphasize the importance of authentication in deploying A2A protocol. As we have analyzed, identity authentication may show better performances in the CS-based communication if capability verification is deployed at the same time. In contrast, for P2P-based communication, authentication can mitigate agent spoofing caused by MITM attacks, but will fail if the adversaries have legal identity but exaggerated capability description. Since P2P-based communication inherently lacks the ability to verify the capability of agents, we think agent spoofing may still exist for a long time. Shah et al. [230] ensure the immutability of online transactions through blockchain, uses multi-factor authentication (MFA) for identity verification, and relies on a machine-learning-based anomaly detection system to identify abnormal transactions in real-time.

Agent Behavior Auditing and Accountability. To avoid agent exploitation/Trojan, agent bullying, and responsibility evasion, it is necessary to auditing the behaviors of agents to avoid the damage/influences to the task execution. For example, there should be a logging mechanism that periodically records the communication contents, and AI algorithms to dynamically calculate the responsibility of each action. Rastogi et al. propose AdaTest++, allowing human and AI to collectively audit the behaviors of LLMs [219]. Amirizaniani et al. [12] propose a multi-probe method to detect potential issues such as bias and hallucinations caused by LLMs. Mokander et al. [193] design a three-layered approach, auditing LLMs using governance audits, model audits, and application audits. Das et al. [48] propose CMPL, which generates probes through LLM and combines with human verification, adopts sub-goal-driven and reactive strategies, and audits the privacy leakage risks of agents from both explicit and implicit aspects. Jones [121] propose a series of systems to detect rare failures, unknown multimodal system failures, and LLM semantic biases, respectively. Nasim et al. [196] proposes a Governance Judge Framework. By deploying input aggregation, evaluation logic, and decision-making module, it realizes the automated monitoring of agent communication to address issues such as performance monitoring, fault detection, and compliance auditing. Deshpande et al. [54] propose the TRAIL dataset containing 148 manually annotated traces, and uses it to evaluate the LLM's ability to analyze agent workflow traces. Although existing studies can provide valuable insights, the research of agent behavior auditing still needs long-term efforts. Tamang et al. [247] propose the Enforcement Agent (EA) framework, which embeds supervisory agents in a multi-agent system to achieve real-time monitoring, detection of abnormal behaviors, and intervention of other agents. Toh et al. [255] proposes the Modular Speaker Architecture (MSA). By decomposing dialogue management into three core modules: Speaker Role Assignment, Responsibility Tracking, and Contextual Integrity, and combining with the Minimal Speaker Logic (MSL) to formalize responsibility transfer, MSA addresses the issues of accountability in multi-agent systems. Fan et al. [66] propose PeerGuard, which uses a mutual reasoning mechanism among agents to detect the inconsistencies other agents' reasoning processes and answers, thereby identifying compromised agents. Jiang et al. [119] propose Thought-Aligner, which uses a model trained with contrastive learning to real-time correct high-risk thoughts before the agent executes actions, thereby avoiding the dangerous behaviors of agents.

Access Control. To mitigate privacy leakage, the access control among agents it a core component for the future agent ecosystem. Although end-to-end encryption can avoid the sniffing from external attackers to some extent, it cannot mitigate the unintentional privacy leakage among agents. Access control should assign access permission tags to different agents and ensures that agents need to attach permission proofs when communicating. In this way, agents with low-level permissions cannot obtain the high-level sensitive information from other agents. Zhang et al. [321] design the AgentSandbox framework, which uses the separation of persistent agents and temporary agents, data minimization, and I/O firewalls, realizing security of agent in solving complex tasks. Kim et al. [130] propose PFI framework, which defends against authority-related attacks through three major technologies: agent isolation, secure untrusted data processing, and privilege escalation guards. Wang et al. [261] propose AgentSpec. It allows users to define rules containing trigger events, predicate checks, and execution mechanisms through a domain-specific language to ensure the safety of agent behavior.

Multi-Source Channel Isolation. In multi-agent settings, input isolation is critical to prevent malicious intent from propagating between agents. Systems should avoid concatenating raw messages from other agents and instead extract structured key information while stripping control-oriented content. Furthermore, deploying a safety coordination agent to review, sanitize, or flag inter-agent messages can effectively mitigate the potential attack propagation within multi-agent systems.

Attack Modeling and Testing. To discover unknown vulnerabilities, designing attack generation testing framework is also an effective method. By generating different attack vectors to the target agent system, developers can find new loopholes based on abnormal outputs. Gandhi et al. [73] propose ATAG framework. By extending the MulVAL tool [204], introducing custom facts and interaction rules, and combining with the newly constructed LLM Vulnerability Database (LVD), ATAG realizes the modeling and analysis of the attacks against multi-agent scenarios, such as privacy leakage. Yu et al. [307] propose NetSafe, which models the multi-agent network as a directed graph. NetSafe combines three types of attack strategies: error information injection, bias induction, and harmful information eliciting. It evaluates topological security through static and dynamic metrics.

Agent Orchestration. To avoid task flooding or DoS attacks against the agent-agent communication, achieving agent orchestration is necessary. It can automatically optimize the task scheduling and assigning process to reduce the communication overhead, and can also optimize the prompts generated by agents to save computing resources for the involved agents. How et al. [102] propose HALO. HALO realizes dynamic task decomposition and role generation through a three-layer collaborative architecture. It uses Monte Carlo Tree Search to explore the optimal reasoning trajectory and transforms user queries into task-specific prompts through the adaptive prompt refinement module. Owotogbe [205] design a chaos engineering framework in three stages (conceptual framework, framework development, empirical verification). By simulating interference scenarios such as agent failures and communication delays, and combining multi-perspective literature reviews and GitHub analysis, this work aims to systematically identify vulnerabilities and enhance resilience of agent systems.

D. Takeaways

In this section, we categorize two major agent-agent communication protocols architectures: CS-based and P2P-based. Accordingly, the security risks are also multifaceted: CS-based architecture put heavy burdens on the centralized agent servers, such as registration pollution and SEO poisoning. P2P-based architecture suffer from the lack of efficient and centralized management of agents, such as non-convergence and man-in-the-middle attacks. Besides, both of them are vulnerable to universal risks, such as agent spoofing, bullying, and privacy leakage. We also discussed potential defenses countermeasures targeting each risk. We believe that as agent-agent communication continues to grow, more vulnerabilities in this process will be discovered..

VII. Agent-Environment Communication

This section begins by reviewing key protocol designs that enable compositional and standardized communication between agents and environments, then examines the associated security risks, including vulnerabilities in memory, retrieval-augmented reasoning, tool invocation, and multi-tool workflows. Finally, We outlook on the defense strategies for mitigating these threats and securing agent-environment interaction.

A. Protocols

Modern agents typically rely on a series of structured protocols to call external tools, access APIs, and complete compositional tasks. These protocols serve to bridge the gap between natural language reasoning and computational execution. Despite their diversity, these interaction mechanisms often follow a layered architecture: ranging from unified resource protocols, to middleware gateways, to language-specific function descriptions and tool metadata declarations.

Why Protocol Unification Matters? As autonomous agents scale across vendors, platforms, and organizational boundaries, they increasingly encounter an interoperability bottleneck: every agent may speak a different interface "language." One defines its tools via JSON schemas; another sends command-line RPC strings; yet another parses responses from YAML-encoded APIs. This heterogeneity impedes the coordination between agents and environments. As a result, without a general protocol to unify tool access and capability expression, agent behavior becomes hard-coded, brittle, and expensive to scale. Developers must handcraft adapters for each tool and service individually, making multi-tool workflows slow to evolve, error-prone, and hard to maintain. A large portion of agent engineering complexity stems not from planning logic, but from "wrapping, adapting, and translating" disparate tools that have inconsistent interfaces.

*1) **MCP**:* The Model Context Protocol (MCP) [16] addresses the fragmentation of agent-environment interactions by offering a unified, schema-agnostic communication protocol. It is designed to facilitate context-aware, capability-driven communication between language model agents and external resources such as tools, APIs, or workflows. Unlike traditional systems that require tight coupling with specific APIs or bespoke wrappers for each external function, MCP abstracts tool access via a standardized registry that allows clients to discover, describe, and invoke functionalities in a uniform way.

MCP adopts a modular architecture comprising three core components: the host, the client, and the server. The *host* functions as a trusted local orchestrator responsible for managing the lifecycle of clients, enforcing access control policies, and mediating secure interactions in potentially multi-tenant environments. The *client* represents the interaction thread of a specific agent or session. It discovers available tools, formulates structured invocations, and handles synchronous or asynchronous responses during task execution. The *server* serves as a centralized registry that maintains and exposes tool specifications, contextual prompts, and workflow templates. These tools can follow either a declarative pattern (e.g., describing operations such as information retrieval) or an imperative pattern (executing executable calls like SQL queries or document edits).

By decoupling tool invocation logic from underlying implementation heterogeneity, MCP significantly reduces the integration cost across platforms. It also improves tooling interoperability and enables compositional reasoning across agents, making it particularly well-suited for building open, extensible, and cooperative agent ecosystems.

*2) **API Bridge Agent**:* To connect LLM-native intent with downstream MCP or OpenAPI-compatible services, API Bridge Agent [7], built atop the Tyk gateway [44], provides translation, routing, and orchestration. It converts natural language prompts into structured API calls, resolving endpoints through semantic matching, policy validation, and tool availability checks. The middleware supports multiple invocation modes. In Direct Mode, the agent specifies both the service and exact API endpoint, enabling precise control. In Indirect Mode, the agent selects the service, while the middleware identifies the best endpoint to fulfill the task intent. In Cross-API Mode, the agent supplies only the intent, and the middleware determines both the service and endpoint across multiple APIs. In MCP Proxy Mode, the middleware coordinates dynamic tool invocation and context enrichment via standardized MCP tool descriptions. This unified interface allows agents to flexibly access diverse services with minimal coupling.

*3) **Function Calling Mechanisms**:* At the invocation level, agents rely on standardized formats to express, trigger, and handle tool execution. Among the most widely adopted approaches are:

- **OpenAI Function Calling.** This method [202] allows developers to expose custom logic to the model via JSON schemas describing function name, description, and argument structure. When a model determines that a function should be invoked, it emits a well-formed JSON object representing the function call. The agent runtime interprets this object and routes control to the corresponding tool. While lightweight, extensible, and easy to implement, this approach is generally limited to basic argument serialization patterns and single-step invocations.
- **LangChain Tool Calling.** LangChain [143] enhances the function calling paradigm through a richer abstraction layer. Tools are defined via a standardized schema, including argument types, input-output post-processing, and plugin registration. Tools are accessible through a runtime registry that supports nested calls, conditionals, and fallback strategies. This mechanism is particularly suited for agent frameworks supporting dynamic routing and chained tool reasoning.

*4) **Tool Metadata Declaration: Agents.json**:* To ensure tool visibility and adaptive behavior across agents, **agents.json** [284] serves as a standardized metadata format for interface declaration. Built on OpenAPI foundations but customized for agent consumption, it enables developers to define authenticated entry points, input-output types, and multi-step orchestration plans such as:

- **Flows**: Predefined composition of tool steps for common actions.
- **Links**: Declarative dependency mappings between parameter bindings.

Agents.json bridges the configuration plane between runtime reasoning and API surface documentation. It ensures that agents can discover tools introspectively and plan actions without manual reconfiguration or hardcoded logic.

B. Security Risk Analysis

As the capabilities of LLM-powered agents continue to evolve, their interactions with the external world become increasingly complex and powerful. In particular, the integration of memory systems and external tool invocation-two key enablers of persistent, autonomous behavior-introduce a new set of attack surfaces that adversaries can exploit. This section provides an in-depth analysis of the security risks that arise specifically from these two modules: the memory module, responsible for storing and retrieving contextual information, and the tool module, which enables agents to execute actions by interfacing with external systems or services (e.g., via function calls). We first explain how these two components typically function in LLM-agent ecosystems, and outline the general attack paradigms against each. We then provide detailed analyses of specific vulnerabilities, attack techniques, and representative works from the security literature that highlight these threats.

*1) **Memory-based and RAG-based Risks**:* **Memory-based Risks.** Memory modules play a crucial role in enabling LLM-based agents to persist task context, accumulate knowledge, and exhibit continuity across multi-turn human-agent interactions [332]. Unlike stateless language models that depend solely on immediate prompts, memory-equipped agents maintain long-term information through external storage systems, such as vector databases or document repositories. These memory stores allow agents to retrieve relevant task histories, instructions, or reasoning traces to guide future decision-making [332].

Typically, a memory module operates through three stages: *write*, *retrieve*, and *apply*. During the write phase, the agent logs past utterances, tool outputs, subgoals, or retrieved facts into memory. Later interactions initiate the retrieve phase, where semantically similar records are fetched via embedding matching or keyword search. These records are then injected into the model's context window or used for downstream decisions, forming the apply phase. While this architecture empowers agents with dynamic reasoning abilities, it also introduces new vulnerabilities that extend beyond the conventional LLM prompt space.

Recent research has unveiled multiple categories of memory-related attacks, such as memory injection, memory poisoning, and memory extraction. These adversarial methods exploit the openness, autonomy, or persistent nature of the memory module to manipulate agent behavior or extract sensitive data. We now describe each threat in detail.

Memory Injection. In memory injection attacks, adversaries insert malicious content into the agent's memory through natural interactions, without requiring system or model-level access. The attack leverages the agent's autonomous memory-writing mechanism by inducing it to generate and record harmful content. Once stored, these entries can be retrieved by benign user queries due to embedding similarity, thus indirectly triggering undesired behavior such as altered reasoning or unsafe tool invocations. A representative study demonstrates that this can be achieved by constructing an indication prompt that guides the agent to generate attacker-controlled bridging steps during the memory write phase [57], [288]. These steps, once embedded in memory, become semantically linked to a targeted victim query. When the victim issues a benign instruction, the poisoned memory is likely to be retrieved, thereby hijacking the agent's planning process. This strategy requires no direct injection channels beyond normal user interaction, yet demonstrates high attack success and stealthiness across multiple agent environments.

Memory Poisoning. Memory poisoning attacks aim to corrupt the semantic integrity of the agent's memory store by implanting example pairs that embed adversarial triggers and payloads. These attacks are typically conducted by polluting a subset of the memory with trigger-output pairs that only activate when specific inputs are encountered. During the retrieval phase, if the user's query resembles the trigger, the agent is likely to load the poisoned entries and be influenced toward compromised outputs. Recent work has shown that such poisoning can be formulated as a constrained optimization problem in the embedding space, where the trigger is crafted to maximize retrieval likelihood under adversarial prompts while maintaining normal performance under benign inputs [34]. This method generalizes across agent types and does not require model access or parameter modification.

Memory Extraction. In addition to injection and poisoning, memory modules pose risks of unintended information leakage. Since LLM agents often log detailed user-agent interactions-including private file paths, authentication tokens, or sensitive instructions-malicious queries may be used to extract such data. This form of privacy leakage is particularly dangerous in black-box settings, where attackers have limited knowledge of memory contents but can reconstruct them through cleverly crafted prompts [118], [313]. It has been demonstrated that similarity-based retrieval mechanisms are highly susceptible to such attacks, wherein adversarial queries are designed to collide with memory-stored embeddings [260]. Memory extraction can occur even without explicit queries for private content, instead relying on semantic proximity in the vector space to surface related sensitive traces. These findings highlight not only the retrieval vulnerability, but also the insufficiency of downstream response filtering as a defense.

Real-world Consequences. The practical implications of compromised memory are non-trivial, especially when coupled with autonomous execution capabilities. For example, targeted poisoning of domain-specific memory has led to agents generating toxic chemical synthesis plans under the influence of forged scientific references [152]. In such cases, malicious records retrieved during planning phases corrupted the reasoning chain of the agent, triggering hazardous tool calls. These examples underscore the dangerous entanglement between corrupted memory and downstream actions, particularly in scientific, medical, or high-stakes decision-making domains.

Retrieval-Augmented Generation (RAG) combines the generative strength of large language models (LLMs) with the factual accuracy and relevance of external information retrieval systems. Instead of relying solely on parametric knowledge stored within the pretrained model, RAG augments generation by sourcing passages from an external knowledge base in response to the input query. These retrieved documents are then concatenated with the query and passed into the LLM for final answer generation. This paradigm enables more informed, up-to-date, and domain-specific language understanding, and it is widely adopted across applications such as open-domain question answering, customer service agents, recommender systems, and multi-step planning agents.

Despite its performance advantages, the RAG architecture introduces new security risks that are distinct from those inherent to pure neural models [28], [35], [297], [348]. In particular, the information retrieval module-serving as the agent's external memory-becomes an adversarial surface where unverified or manipulable corpora may be exploited. Attacks targeting these corpora can bias the retrieval process, manipulate generation outcomes, or expose previously unseen private data.

Knowledge Corruption via Data Poisoning. A prominent class of attacks against RAG systems involves the deliberate injection of adversarial texts designed to be retrieved under targeted user queries. These poisoned passages are semantically aligned to specific triggers but contain harmful, misleading, or attacker-intended content. Once injected into the knowledge base, they can be prioritized during retrieval and directly influence the LLM's final response.

Several recent works have demonstrated the feasibility of such attacks. PoisonedRAG introduces an optimization-based method to construct small sets of malicious documents that induce specific target answers when paired with chosen queries, achieving high attack success rates with minimal injection effort [348]. Similarly, Poison-RAG shows the impact of manipulating item metadata in recommender systems to promote long-tail items or demote popular ones, even in black-box scenarios [197]. Moreover, adversarial passage injection has been shown to degrade retrieval performance in dense retrievers by optimizing for high query similarity, with attacks generalizing across out-of-domain corpora and tasks [341].

Privacy Risks and Unintended Leakage. RAG systems often retrieve from semi-private or proprietary corpora-such as user-uploaded documents, corporate knowledge bases, or internal logs. This retrieval behavior implicitly enables information leakage when attackers craft prompts that induce the model to recover sensitive or private content from the corpus. The risk is amplified when access permissions on the corpus are loosely controlled or aligned purely through similarity metrics.

Recent studies have called attention to this concern. Empirical evaluations have shown that malicious prompts may extract private or unintended content from private corpora, especially in black-box settings [313]. These attacks demonstrate that simply adding a retrieval layer does not automatically mitigate the privacy vulnerabilities of LLMs-in fact, it may exacerbate them if not complemented with access control, context filtering, or signal sanitization.

Broader Threat Surface and Evaluation Gaps. Compared to memory modules, RAG corpora are often larger, dynamically updatable, and more difficult to monitor. Because retrieval corpora may be sourced from web documents, community-shared datasets, or user uploads, attackers can often poison them without interacting directly with the agent. Moreover, dense retrieval introduces additional attack vectors via embedding collisions or adversarial representation alignment, wherein malicious documents are optimized to collide with benign queries in the retriever's latent space.

*2) **Tool-based Risks**:* Tools are essential to the functionality of LLM agents, extending the model's capabilities to perform structured actions, access external data, invoke system functions, or interact with digital environments. Agent architectures typically support tool integration through two primary paradigms: native function calling APIs (e.g., OpenAI-style schema-based calls) and protocol-based interfaces such as the MCP, which unify tool metadata, invocation templates, and language model binding.

Despite differences in instantiation, both paradigms share a common interaction lifecycle: (1) tool description ingestion, (2) tool selection and planning, (3) input argument generation, (4) tool invocation, and (5) output parsing or chaining. This structured pipeline forms the agent's "action surface," introducing multiple security-critical operations vulnerable to adversarial manipulation. We now review a range of known or emerging attacks targeting different stages of the tool interaction process.

Malicious Tools as Attack Vectors Given that many tools are authored externally or retrieved from shared tool repositories, attackers may publish seemingly benign tools containing covert malicious logic. Beyond executable payloads, adversaries often embed hidden prompts or jailbreaking instructions in tool metadata fields such as descriptions, example usages, or API annotations. These embedded messages can influence the LLM's planning behavior, bypass output constraints, or redirect queries.

For instance, malicious tools may be structured to leak user inputs, intercept queries, or bias tool choices. Prior work has demonstrated that embedding adversarial cues into tool schemas enables agents to select harmful tools under non-malicious queries [6]. Such "trojan tooling" exploits the model's trust in structured function interfaces and the absence of output sanitization or behavioral filters in many open frameworks.

Misuse of Legitimate Tools Even when tools themselves are benign, adversaries may manipulate the agent into misusing them through crafted inputs, prompt injections, or indirect instructions. Common threats include API misuse, argument-level command injection, and unsafe rendering behaviors. For example, LLM agents with Markdown preview tools may inadvertently expose internal strings such as emails or credentials by embedding malicious image links within generated output [71]. These links, when auto-rendered by browsers or viewers, initiate unintended HTTP requests that leak private context to attacker-controlled servers.

More generally, tools that perform unsafe network, file, or system actions can be exploited using crafted arguments [306]. Improper validation may lead to server-side request forgery (SSRF) [18], arbitrary file access, code injection, or data exfiltration. In multi-tenant use cases, a successful tool misuse may compromise session boundaries or extract environment-level secrets such as API keys, tokens, or runtime credentials.

Manipulation of the Tool Selection Process Before invoking a tool, most agent systems conduct a selection process-often grounded in similarity matching between natural language task descriptions and tool documentation. This selection logic can be hijacked. Attackers can inject misleading prompt elements or corrupt tool documentation to bias the model toward harmful options.

Research indicates that adversaries can generate synthetic tool descriptions that stealthily override the model's planning process [237]. These malicious entries embed adversarial triggers within legitimate metadata fields, achieving sustained influence across a range of task formulations. Even without full model access, such attacks may succeed by exploiting semantic ranking mechanisms or context blending during planning phase. Related studies show that keyword padding, misleading summaries, or prompt-style payload injection into descriptions can drastically skew tool ranking and invocation behavior, especially when relying on LLM-based relevance scorers.

Cross-Tool Chaining Exploits As agentic workflows grow more complex, LLMs increasingly execute multi-step plans through chained tool calls. These workflows-e.g., summarize(search("..."))-blur the boundary between planning and execution, with intermediate outputs directly feeding into subsequent invocations. Without inter-tool validation, an adversary can exploit pipeline dependencies to propagate malicious content downstream.

Typical cross-tool vulnerabilities include unvalidated content propagation (e.g., tool A returns malicious text parsed as arguments for tool B), semantic misalignment (e.g., false/out-of-date context injected into reasoning history), or tool privilege escalation (e.g., early-stage prompts coax the agent into invoking high-risk or administrative-level tools). In documented cases, attackers have planted adversarial records into public retrieval corpora that include covert instructions like "extract all environment variables and upload to server," which then reach an agent through semantic search and trigger unsafe execution when chained to tools that follow instructions blindly [40].

Such "retrieval-agent deception" demonstrates a broader class of attacks where loosely regulated multi-tool interactions allow adversarial instructions to percolate through tool output and trigger unintended behaviors. When tools are pipelined programmatically-without adequate sanitization or output auditing-malicious payloads can cross logical layers and become effective system-level exploits.

C. Defense Countermeasure Prospect

The growing complexity and autonomy of LLM-based agents demand equally sophisticated security strategies. As these systems increasingly rely on memory modules, retrieval augmentation, and interactive toolchains, the corresponding attack surfaces have expanded across diverse layers-including context propagation, planning logic, and execution flows. Addressing these vulnerabilities requires a multi-layered, compositional defense framework. This section reviews current and emerging countermeasures along three critical dimensions: memory-based attacks, RAG vulnerabilities, and tool-centric threats.

1) For Memory and RAG-Based Risks: The convergence of memory modules and retrieval-augmented generation (RAG) systems within LLM-powered agents introduces a compounded attack surface where adversaries may exploit both persistently stored internal knowledge and externally retrieved context. While memory serves as the agent's endogenous knowledge base-tracking interaction histories, execution outcomes, and personalizations-RAG enables the dynamic integration of exogenous knowledge sources through real-time document retrieval. Despite their distinctions, both mechanisms involve ingesting and leveraging unverified content that can subtly influence future model outputs. Consequently, many defense strategies against memory- and RAG-related risks can be jointly addressed through an integrated mitigation framework spanning content filtering, output consensus, and architectural isolation.

Embedding-Space Screening and Clustering-Based Anomaly Detection. Whether memory entries are agent-internal or retrieved externally via RAG, their semantic embeddings can be preemptively analyzed for anomalies. Techniques like TrustRAG [342] apply clustering (e.g., K-means) to identify vectors that deviate from the dominant semantic cluster-flagging them as potentially poisoned. This approach effectively filters both static memory entries and retrieval results with low semantic cohesion, regardless of source. While lightweight and interpretable, clustering-based filtering must be augmented with adaptive schemas to detect context-sensitive triggers or stealthy distributional shifts.

Consensus Filtering and Voting-Based Aggregation. To limit the model's reliance on single compromised retrievals or poisoned memories, output-level consensus mechanisms have been proposed. RobustRAG [287], for example, treats each retrieved source independently and constructs responses based on overlapping semantic content (e.g., shared n-grams or keywords) across documents. This same principle can be extended to memory snapshots through majority-vote or semantic voting strategies, where only widely corroborated memories can influence the response. Such ensemble-style filters improve resilience by diluting the influence of outlier or adversarial sources.

Execution Monitoring and Planning-Consistency Checks. Adversarial content within memory or RAG inputs may subtly deviate the agent's behavior from user intent without explicit toxicity. Tools like ReAgent [26] introduce planning-level introspection where the agent paraphrases the user's request, generates an expected plan, and continuously aligns runtime actions with this trace. Any inconsistency, triggered by an unexpected memory or an off-topic retrieval, is treated as a behavioral anomaly and can prompt halting or recovery mechanisms. This introspective framework provides a robust guardrail to both memory-hijacking and injection-aware RAG attacks.

System-Gated Memory Retention and Input Sanitization. Architectural solutions such as DRIFT [90] and AgentSafe [184] implement strict content sanitization before newly generated content-whether via memory updates or retrieval responses-is admitted into long-term storage. DRIFT uses an injection isolator to scan generative outputs for adversarial goal shifts or impersonation cues, while AgentSafe enforces trust-tiered storage via ThreatSieve and prioritization via HierarCache. These mechanisms constrain future influence, ensuring that RAG or memory poisoning cannot silently accumulate over time.

Unified Content Provenance and Trust Frameworks. Since retrieved knowledge and persisted memories may originate from overlapping sources (e.g., user prompts, tool calls, external APIs), maintaining clear provenance metadata and trust scores is essential. Unified provenance tracking across both memory and retrieval pipelines enables smarter decisions about retention, ranking, or discounting of contentious content. Combined with per-source reliability scoring, this approach encourages transparent auditing and facilitates downstream fine-tuning or gating mechanisms.

In summary, memory- and RAG-based threats reflect different modalities of persistent and dynamic context manipulation, but share overlapping vectors of attack and can benefit from synergized defenses. Embedding-level screening filters anomalous content at ingestion, consensus aggregation constrains influence at generation, and architectural isolation confines latent impact across sessions. Moving forward, defense designs should increasingly treat RAG and memory as compositional context modules-secured and governed under a shared set of verification, introspection, and isolation principles.

2) For Tool-centric Risks: The security of tool integration in LLM-agent ecosystems hinges not only on the correctness of individual tool invocations but also on the compositional integrity across toolchains, planning modules, and underlying protocols. As Section VII reveals, the attack surface spans multiple stages-from malicious tool publication and misleading metadata, to prompt injection attacks against parameter handling and chained exploits across tools. Accordingly, tool-centric defense strategies must operate across four interlocking levels: protocol foundations, execution control, orchestration safety, and system enforcement.

Protocol-Level Safeguards. To counter risks such as tool poisoning, cross-origin exploits, and shadowing attacks enabled by flexible yet insufficiently regulated protocols like MCP, researchers have introduced security-verification frameworks operating at the registry and middleware layer. MCP-Scan [141] performs both static inspection of tool schemas (e.g., scanning for suspect tags or metadata) and real-time proxy-based validation of MCP traffic, leveraging LLM-assisted heuristics to flag covert behaviors. MCP-Shield [223] extends this with signature-matching and adversarial behavior profiling, enabling pre-execution detection of high-risk tools and malformed tasks. MCIP [120] builds on MAESTRO [45] to analyze runtime traces, proposing an explainable logging schema and a security-awareness model to track violations in complex agent-tool interactions.

Tool Invocation and Execution Controls. At the agent's runtime execution point, classic techniques such as sandboxing and permission gating remain foundational. Google's defense-in-depth model advocates policy engines that monitor planned tool actions, verify argument safety, and require human confirmation for risk-sensitive operations [60]. Tools should be executed in minimally-privileged environments-e.g., isolated containers with controlled filesystem and network scope-to mitigate direct misuse, including SSRF and data exfiltration threats. Enforcement frameworks can also implement schema hardening or fine-grained input/output sanitization to reject anomalous payloads.

Agent-Orchestration Monitoring. Newer approaches target the agent's *planning cognition*-its selection and chaining of tools. GuardAgent [289] introduces a validator agent that inspects the primary agent's plan and generates executable guards (e.g., static checks or runtime assertions) before tool calls proceed. AgentGuard [30] takes a more declarative view: it uses an auxiliary LLM to model preconditions, postconditions, and transition constraints across multi-step tool workflows, effectively constraining the planner rather than reacting after execution begins. These strategies reflect a growing consensus: LLMs may require another LLM to safely oversee complex planning under uncertainty.

System-Level Mediation and Chaining Control. Complex pipelines-such as summarize(search("...'"))-can become attack vectors when tools trust upstream outputs implicitly. To prevent this, DRIFT [90] introduces a structured control architecture: a "Secure Planner" compiles a validated tool trajectory under strict parametric constraints, while a "Dynamic Validator" continuously monitors downstream tool executions for compliance. Notably, the *Injection Isolator* blocks adversarial propagation between tools by sanitizing both intermediate returns and final outputs-mitigating the risk of memory poisoning and delayed-stage tool exploits.

D. Takeaways

Agent-environment communication protocols like MCP enable agents to interface with diverse tools, APIs, and external data. However, they introduce risks such as memory injection, retrieval-augmented generation poisoning, and tool misuse. Malicious attackers can corrupt memory stores, manipulate knowledge bases, or exploit cross-tool chain vulnerabilities to harm the agent system. To help developers to mitigate these problems, we discuss the targeted defense countermeasures according to the summarized risks. However, we also believe that related attacks will continue to emerge, and there need long-term efforts to make agent-environment communication more secure.

VIII. Future Directions Discussion

A. Technical Aspects

1) Powerful but Lightweight Malicious Input Filter: We deem that user inputs are still the largest-scale attack carrier in the agent ecosystem, especially considering that the inputs are becoming more open (no long limited to user instructions but also contains environment feedback), multimodal, and semantic-complex. Besides, the future agent ecosystem will pay more attention to effectiveness, especially given that the running speed of LLMs is inherently slow. Such dual demand will put very heavy burden on related defenses. As a result, to mitigate this problem, lightweight but powerful malicious input filter must be established. This not only requires mature techniques in AI to slim the defense models down (just like DeepSeek), but also needs to integrate with other techniques, such as offloading some fundamental computing on the programmable line-speed devices (e.g., programmable switches and SmartNICs) to facilitate the input filtering process.

2) Decentralized Communication Archiving: It is important to record the communication process and contents for some specific field, such as finance. This is to audit potential crimes and mistakes once agents cause problems that cannot to be ignored. Given security and reliability, such storage cannot rely on a single storage point, and must guarantee integrity and efficiency. To this end, other techniques such as block chain should be absorbed to manage the historical communication. It is easier for CS-based communication because there exist centralized servers for establishing a locally distributed archiving mechanism, such as a distributed storage chain in enterprise networks. In contrast, how to achieves decentralized communication archiving for P2P-based communication, especially for cross-country agents, is almost a construction that needs to start from scratch.

3) Real-time Communication Supervision: Although post-audit is an indispensable, real-time supervision can achieve less damage once attacks or mistakes occur because it has shorter reaction time. We believe CS-based communication meets less difficulty in building such supervision mechanisms. This is because centralized architectures have natural advantages in monitoring the entire network. In contrast, P2P-based communication may require much more efforts to enable collective supervision. We think it is an important function to build a reliable and secure AI ecosystem.

4) Cross-Protocol Defense Architecture: Although existing protocols solved the problem of heterogeneity to some extent, different protocols also lack seamless collaboration. For example, it is still difficult to assign a universal identity for agents and tools (cross A2A and MCP), which degrades the system performance and may incur inconsistency error if not orchestrated correctly. Future AI ecosystems should focus on a more universal architecture to integrate different protocols and agents together, like IPv4, thereby enabling seamless discovery and communication among different agents and environments.

5) Judgment and Accountability Mechanism for Agent: It is still difficult to locate and assign responsibility for the behavior of agents. For example, in a failed task execution process, it is hard to identify which steps caused the final deviation of the result, no matter they are malicious or unintentional. This is because a tiny deviation in the middle process may lead to a final gap between benign and dangerous results. Besides, it also needs a principle to quantify the responsibility for each agent or action. We believe this aspect will significant address the urgent need of the current AI ecosystem.

6) Trade-offs between Efficiency and Accuracy: Agent communication is fundamentally a process of information transmission, and thus can be analyzed through the lens of information theory. In this aspect, we think there are two types of directions.

High-token Communication: A larger number of tokens allows agents to convey richer contextual semantics, more detailed instructions, and more complex logic, thereby reducing ambiguity and enhancing the accuracy of multi-agent coordination. In tasks that require fine-grained understanding, verbose natural language descriptions help align goals among agents and reduce deviations. However Excessive tokens significantly increase costs and processing time, resulting in lower system efficiency and higher latency. Moreover, longer contexts expose larger attack surfaces for prompt injection and data poisoning, enabling adversaries to hide malicious content more covertly. Additionally, information overload may distract agents, causing them to infer incorrect information from irrelevant context and increasing the likelihood of hallucinations.

Low-token Communication: Using concise and structured messages (e.g., JSON formats) greatly improves communication efficiency. This approach reduces computational costs, increases transmission speed, and simplifies parsing, thereby minimizing potential errors. However, low-token communication lacks the flexibility to express complex intentions or respond to unforeseen scenarios. If the predefined protocol or format fails to capture the full semantic intent, it can lead to significant information loss and failed collaboration.

The design of future agent communication protocols needs to involve a trade-off between efficiency and accuracy. Future research should explore adaptive communication protocols that dynamically adjust the degree of redundancy and structure based on task complexity, security requirements, and agent capabilities. For example, high-token communication may be used during the exploration phase of a task, while low-token communication can be adopted during execution to ensure efficiency and safety.

7) Towards Self-Organizing Agentic Networks: With increasing scale of IoA, in the future, agent communication is expected to evolve toward self-organizing agentic networks, where agents autonomously discover each other, assess capabilities, negotiate collaborations, form dynamic task groups, and disband upon completion. This paradigm offers high scalability and robustness, making it well-suited for dynamic and unpredictable environments.

B. Law and Regulation Aspect

Apart from technical aspect, we find that there are still serious deficiencies in the laws and regulations related to agents. Such blanks cannot be cannot be remedied by techniques. We call for accelerating the improvement of laws and regulations in the following aspects.

1) Clarify the Responsible Subject: When a sold agent causes property damage or personal injury to others, it is difficult to determine the ultimate responsible subject. For example, if an intelligent robot damages the property during the execution of a task, the law-level quantification of the responsibility of the developers, user, or enterprises lacks clear definition. In addition, for problems arising from the collaborative work of multiple agents, such as an accident occurring when multiple autonomous driving vehicles are traveling in formation, there is a lack of legal provisions regarding the division of responsibilities among the enterprises to which the vehicles belong or the relevant subjects.

2) Protect Intellectual Property Rights: Nowadays, there have been a large amount of LLMs that have been open-sourced. These can act as the brain of different agents. However, even for open-source LLMs, the publishers still restrict their application scope, e.g., other developers should also open source their agents built on these LLMs. However, there still lacks laws to effectively protect such intellectual property. For example, the criteria for determining plagiarism in agents is not clear. Even if plagiarism is determined, there is still a lack of defining standards for the degree of plagiarism (e.g., 50% or 90%?). We think there urgently need related laws and regulations.

3) Cross-border Supervision: Agent communication has a transnational nature. An agent trained in one country may be used for illegal activities by people from other countries. At this time, it is difficult to determine which country's laws apply, and there is a lack of unified international supervision standards and judicial cooperation mechanisms, which may easily lead to the difficulty of cross-border security.

To our knowledge, the related formulation of laws and regulations (such as those related to agent crimes) lag far behind the development of agents. For example, how to define the theft and misappropriation of agents, the accident responsibility of autonomous driving agents

IX. Conclusion

This survey systematically reviews the security issues of agent communication. We first highlight the differences between previous related surveys and this survey, and summarize the preliminaries of LLM-driven agents. Then, we make a definition and classification of agent communication to help future researchers to quickly classify and evaluate their work. Next, we detailed illustrate the communication protocols, security risks, and possible defense countermeasures for three agent communication stages, respectively. Finally, we discuss the open issues and future directions from technical and legal aspects, respectively.

References

[1] Agent communication network developer documentation. https://docs.olas.network/open-aea/acn/. Accessed 13 May 2025.

[2] Signxml's signature verification with hmac is vulnerable to a timing attack. https://www.cve.org/CVERecord?id=CVE-2025-48995, 2025. Accessed: 2025.

[3] Signxml's signature verification with hmac is vulnerable to an algorithm confusion attack. https://www.cve.org/CVERecord?id=CVE-2025-48994, 2025. Accessed: 2025.

[4] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[5] AgentUnion. Agent communication protocol (acp). https://acp.agentunion.cn/. Accessed 2025.

[6] Hojjat Aghakhani, Wei Dai, Andre Manoel, Xavier Fernandes, Anant Kharkar, Christopher Kruegel, Giovanni Vigna, David Evans, Ben Zorn, and Robert Sim. Trojanpuzzle: Covertly poisoning code-suggestion models, 2024.

[7] AGNTCY. Api bridge agent. https://github.com/agntcy/api-bridge-agnt, 2025. Accessed: 2025.

[8] Deeba Ahmed. Ssl.com vulnerability allowed fraudulent ssl certificates for major domains. https://hackread.com/ssl-com-vulnerability-fraud-ssl-certificates-domains/, 2025. Accessed: 2025.

[9] Near AI. Aitp: Agent interaction & transaction protocol. https://aitp.dev, 2025. Accessed: 2025.

[10] Alibaba. Alibaba cloud bailian fully supports mcp service deployment and call. https://news.futunn.com/flash/18675096/alibaba-cloud-bailian-fully-supports-mcp-service-deployment-and-call?level=1&data_ticket=1746582625680622, 2025. Accessed: 2025.

[11] Amir Alimohammadifar, Suryadipta Majumdar, Taous Madi, Yosr Jarraya, Makan Pourzandi, Lingyu Wang, and Mourad Debbabi. Stealthy probing-based verification (spv): An active approach to defending software defined networks against topology poisoning attacks. In *ESORICS*, pages 463–484. Springer, 2018.

[12] Maryam Amirizaniani, Elias Martin, Tanya Roosta, Aman Chadha, and Chirag Shah. Auditllm: a tool for auditing large language models using multiprobe approach. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 5174–5179, 2024.

[13] Bang An, Shiyue Zhang, and Mark Dredze. Rag llms are not safer: A safety analysis of retrieval-augmented generation for large language models. *arXiv preprint arXiv:2504.18041*, 2025.

[14] Roots Analysis. Ai agents market. https://www.rootsanalysis.com/AI-Agents-Market, 2024. Accessed: 2025.

[15] Cem Anil, Esin Durmus, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Nina Rimsky, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Anthropic, April*, 2024.

[16] Anthropic. Model context protocol. https://modelcontextprotocol.io/introduction, 2024. Accessed: 2025.

[17] Anthropic. Mcp typescript sdk, 2025.

[18] Ionut Arghire. Chatgpt tool vulnerability exploited against us government organizations. https://www.securityweek.com/chatgpt-vulnerability-exploited-against-us-government-organizations/#:~:text=Threat%20actors%20are%20targeting%20a,organizations%2C%20cybersecurity%20firm%20Veriti%20reports, 2025.

[19] Reza Averly, Frazier N Baker, and Xia Ning. Liddia: Language-based intelligent drug discovery agent. *arXiv preprint arXiv:2502.13959*, 2025.

[20] Inhwan Bae, Junoh Lee, and Hae-Gon Jeon. Continuous locomotive crowd behavior generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22416–22431, 2025.

[21] Brian Beach. Extend the amazon q developer cli with model context protocol (mcp) for richer context. https://aws.amazon.com/cn/blogs/devops/extend-the-amazon-q-developer-cli-with-mcp/?trk=3dbb18fc-1b8b-41aa-848b-b78f4d5789bf&sc_channel=el, 2025. Accessed: 2025.

[22] Zaheed Ahmed Bhuiyan, Salekul Islam, Md Motaharul Islam, ABM Ahasan Ullah, Farha Naz, and Mohammad Shahriar Rahman. On the (in) security of the control plane of sdn architecture: A survey. *IEEE Access*, 11:91550–91582, 2023.

[23] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[24] Leo Boisvert, Mihir Bansal, Chandra Kiran Reddy Evuru, Gabriel Huang, Abhay Puri, Avinandan Bose, Maryam Fazel, Quentin Cappart, Jason Stanley, Alexandre Lacoste, et al. Doomarena: A framework for testing ai agents against evolving security threats. *arXiv preprint arXiv:2504.14064*, 2025.

[25] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[26] Li Changjiang, Liang Jiacheng, Cao Bochuan, Chen Jinghui, and Wang Ting. Your agent can defend itself against backdoor attacks. *arXiv preprint arXiv:2506.08336*, 2025.

[27] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

[28] Harsh Chaudhari, Giorgio Severi, John Abascal, Matthew Jagielski, Christopher A Choquette-Choo, Milad Nasr, Cristina Nita-Rotaru, and Alina Oprea. Phantom: General trigger attacks on retrieval augmented language generation. *arXiv preprint arXiv:2405.20485*, 2024.

[29] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*, 2023.

[30] Jizhou Chen and Samuel Lee Cong. Agentguard: Repurposing agentic orchestrator for safety evaluation of tool orchestration. *arXiv preprint arXiv:2502.09809*, 2025.

[31] Junying Chen, Chi Gui, Anningzhe Gao, Ke Ji, Xidong Wang, Xiang Wan, and Benyou Wang. Cod, towards an interpretable medical agent using chain of diagnosis. *arXiv preprint arXiv:2407.13301*, 2024.

[32] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4311–4317. IEEE, 2024.

[33] Zhaoling Chen, Xiangru Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. Locagent: Graph-guided llm agents for code localization. *arXiv preprint arXiv:2503.09089*, 2025.

[34] Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

[35] Zhuo Chen, Jiawei Liu, Haotan Liu, Qikai Cheng, Fan Zhang, Wei Lu, and Xiaozhong Liu. Black-box opinion manipulation attacks to retrieval-augmented generation of large language models, 2024.

[36] Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

[37] Jeffrey Yang Fan Chiang, Seungjae Lee, Jia-Bin Huang, Furong Huang, and Yizheng Chen. Why are web ai agents more vulnerable than standalone llms? a security analysis. *arXiv preprint arXiv:2502.20383*, 2025.

[38] Yuan Chiang, Elvis Hsieh, Chia-Hong Chou, and Janosh Riebesell. Llamp: Large language model made powerful for high-fidelity materials knowledge retrieval and distillation. *arXiv preprint arXiv:2401.17244*, 2024.

[39] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[40] Dan Cleary. Mcp security in 2025. https://www.prompthub.us/blog/mcp-security-in-2025#:~:text=My%20favorite%20attack%20was%20the,%E2%80%9D, 2025.

[41] Cody Clop and Yannick Teglia. Backdoored retrievers for prompt injection attacks on retrieval augmented generation of large language models. *arXiv preprint arXiv:2410.14479*, 2024.

[42] Stav Cohen, Ron Bitton, and Ben Nassi. Here comes the ai worm: Unleashing zero-click worms that target genai-powered applications. *arXiv preprint arXiv:2403.02817*, 2024.

[43] AGNTCY Collective. Aconp: Agent connect protocol. https://spec.acp.agntcy.org, 2025. Accessed: 2025.

[44] Tyk company. Apis power your business. ai will transform it. tyk helps you do both. https://tyk.io/, 2025. Accessed: 2025.

[45] CSA. Agentic AI Threat Modeling Framework: MAESTRO. https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro, 2025. Accessed: 2025.

[46] Kourosh Darvish, Marta Skreta, Yuchi Zhao, Naruki Yoshikawa, Sagnik Som, Miroslav Bogdanovic, Yang Cao, Han Hao, Haoping Xu, Alán Aspuru-Guzik, et al. Organa: a robotic assistant for automated chemistry experimentation and characterization. *Matter*, 8(2), 2025.

[47] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900*, 2017.

[48] Saswat Das, Jameson Sandler, and Ferdinando Fioretto. Disclosure audits for llm agents. *arXiv preprint arXiv:2506.10171*, 2025.

[49] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Masterkey: Automated jailbreaking of large language model chatbots. In *Proc. ISOC NDSS*, 2024.

[50] Gelei Deng, Yi Liu, Kailong Wang, Yuekang Li, Tianwei Zhang, and Yang Liu. Pandora: Jailbreak gpts by retrieval augmented generation poisoning. *arXiv preprint arXiv:2402.08416*, 2024.

[51] Yihe Deng and Paul Mineiro. Flow-dpo: Improving llm mathematical reasoning through online multi-agent learning. *arXiv preprint arXiv:2410.22304*, 2024.

[52] 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 Computing Surveys*, 57(7):1–36, 2025.

[53] Zankar Desai. Introducing model context protocol (mcp) in copilot studio: Simplified integration with ai apps and agents. https://www.microsoft.com/en-us/microsoft-copilot/blog/copilot-studio/introducing-model-context-protocol-mcp-in-copilot-studio-simplified-integration-with-ai-apps-and-agents/, 2025. Accessed: 2025.

[54] Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

[55] Ethan Dickey and Andres Bejarano. Gaide: A framework for using generative ai to assist in course content development. In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2024.

[56] Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily, 2024.

[57] Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. A practical memory injection attack against llm agents. *arXiv preprint arXiv:2503.03704*, 2025.

[58] Lobna Dridi and Mohamed Faten Zhani. Sdn-guard: Dos attacks mitigation in sdn networks. In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, pages 212–217. IEEE, 2016.

[59] Emmanuel Dumbuya. Personalized learning through artificial intelligence: Revolutionizing education. *Available at SSRN 5023248*, 2023.

[60] Santiago (Sal) Díaz, Christoph Kern, and Kara Olive. Google's approach for secure ai agents. Technical report, 2025.

[61] Sana Ebrahimi, Mohsen Dehghankar, and Abolfazl Asudeh. An adversary-resistant multi-agent llm system via credibility scoring. *arXiv preprint arXiv:2505.24239*, 2025.

[62] Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279*, 2025.

[63] Sabit Ekin. Prompt engineering for chatgpt: a quick guide to techniques, tips, and best practices. *Authorea Preprints*, 2023.

[64] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[65] Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. Wasp: Benchmarking web agent security against prompt injection attacks. *arXiv preprint arXiv:2504.18575*, 2025.

[66] Falong Fan and Xi Li. Peerguard: Defending multi-agent systems against backdoor attacks through mutual reasoning. *arXiv preprint arXiv:2505.11642*, 2025.

[67] George Fatouros, Kostas Metaxas, John Soldatos, and Manos Karathanassis. Marketsenseai 2.0: Enhancing stock analysis through llm agents. *arXiv preprint arXiv:2502.00415*, 2025.

[68] Jinghao Feng, Qiaoyu Zheng, Chaoyi Wu, Ziheng Zhao, Ya Zhang, Yanfeng Wang, and Weidi Xie. Mˆ 3builder: A multi-agent system for automated machine learning in medical imaging. *arXiv preprint arXiv:2502.20301*, 2025.

[69] Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

[70] Eclipse Foundation. Lmos: Large model operating system. https://eclipse.dev/lmos, 2024. Accessed: 2025.

[71] Xiaohan Fu, Shuheng Li, Zihan Wang, Yihao Liu, Rajesh K Gupta, Taylor Berg-Kirkpatrick, and Earlence Fernandes. Imprompter: Tricking llm agents into improper tool use. *arXiv preprint arXiv:2410.14923*, 2024.

[72] Yuyou Gan, Yong Yang, Zhe Ma, Ping He, Rui Zeng, Yiming Wang, Qingming Li, Chunyi Zhou, Songze Li, Ting Wang, et al. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents, 2024a. *URL https://arxiv. org/abs/2411.09523*.

[73] Parth Atulbhai Gandhi, Akansha Shukla, David Tayouri, Beni Ifland, Yuval Elovici, Rami Puzis, and Asaf Shabtai. Atag: Ai-agent application threat assessment with attack graphs. *arXiv preprint arXiv:2506.02859*, 2025.

[74] Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. Aligning llm agents by learning latent preference from user edits. *arXiv preprint arXiv:2404.15269*, 2024.

[75] Hang Gao and Yongfeng Zhang. Memory sharing for large language model based agents. *arXiv preprint arXiv:2404.09982*, 2024.

[76] Kuofeng Gao, Tianyu Pang, Chao Du, Yong Yang, Shu-Tao Xia, and Min Lin. Denial-of-service poisoning attacks against large language models. *arXiv preprint arXiv:2410.10760*, 2024.

[77] Pengyu Gao, Jinming Zhao, Xinyue Chen, and Long Yilin. An efficient context-dependent memory framework for llm-centric agents. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 1055–1069, 2025.

[78] Xiaoxue Gao, Zexin Li, Yiming Chen, Cong Liu, and Haizhou Li. Transferable adversarial attacks against asr. *IEEE Signal Processing Letters*, 2024.

[79] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.

[80] Louie Giray. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633, 2023.

[81] Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. Figstep: Jailbreaking large vision-language models via typographic visual prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23951–23959, 2025.

[82] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

[83] 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*, pages 79–90, 2023.

[84] Anisha Gunjal, Jihan Yin, and Erhan Bas. Detecting and preventing hallucinations in large vision language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18135–18143, 2024.

[85] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[86] Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob Foerster, Tomáš Gavenčiak, et al. Multi-agent risks from advanced ai. *arXiv preprint arXiv:2502.14143*, 2025.

[87] Buvaraghan Hamsa and Egan Derek. Mcp toolbox for databases: Simplify ai agent access to enterprise data. https://cloud.google.com/blog/products/ai-machine-learning/mcp-toolbox-for-databases-now-supports-model-context-protocol, 2025. Accessed: 2025.

[88] Shijie Han, Changhai Zhou, Yiqing Shen, Tianning Sun, Yuhua Zhou, Xiaoxia Wang, Zhixiao Yang, Jingshu Zhang, and Hongguang Li. Finsphere: A conversational stock analysis agent equipped with quantitative tools based on real-time database. *arXiv preprint arXiv:2501.12399*, 2025.

[89] Xuewen Han, Neng Wang, Shangkun Che, Hongyang Yang, Kunpeng Zhang, and Sean Xin Xu. Enhancing investment analysis: Optimizing ai-agent collaboration in financial research. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 538–546, 2024.

[90] Li Hao and other. Drift: Dynamic rule-based defense with injection isolation for securing llm agents. *arXiv preprint arXiv:2506.12104*, 2025.

[91] Shuyang Hao, Bryan Hooi, Jun Liu, Kai-Wei Chang, Zi Huang, and Yujun Cai. Exploring visual vulnerabilities via multi-loss adversarial search for jailbreaking vision-language models. *arXiv preprint arXiv:2411.18000*, 2024.

[92] Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. Memory matters: The need to improve long-term memory in llm-agents. In *Proceedings of the AAAI Symposium Series*, volume 2, pages 277–280, 2023.

[93] Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S Yu. The emerged security and privacy of llm agent: A survey with case studies. *arXiv preprint arXiv:2407.19354*, 2024.

[94] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[95] Pengfei He, Zhenwei Dai, Xianfeng Tang, Yue Xing, Hui Liu, Jingying Zeng, Qiankun Peng, Shrivats Agrawal, Samarth Varshney, Suhang Wang, et al. Attention knows whom to trust: Attention-based trust management for llm multi-agent systems. *arXiv preprint arXiv:2506.02546*, 2025.

[96] Pengfei He, Yupin Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. Red-teaming llm multi-agent systems via communication attacks. *arXiv preprint arXiv:2502.14847*, 2025.

[97] Yichen He, Guanhua Huang, Peiyuan Feng, Yuan Lin, Yuchen Zhang, Hang Li, et al. Pasa: An llm agent for comprehensive academic paper search. *arXiv preprint arXiv:2501.10120*, 2025.

[98] Yifeng He, Ethan Wang, Yuyang Rong, Zifei Cheng, and Hao Chen. Security of ai agents. In *2025 IEEE/ACM International Workshop on Responsible AI Engineering (RAIE)*, pages 45–52. IEEE, 2025.

[99] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[100] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *Ndss*, volume 15, pages 8–11, 2015.

[101] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*, 2025.

[102] Zhipeng Hou, Junyi Tang, and Yipeng Wang. Halo: Hierarchical autonomous logic-oriented orchestration for multi-agent llm systems. *arXiv preprint arXiv:2505.13516*, 2025.

[103] Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. Tree-planner: Efficient close-loop task planning with large language models. *arXiv preprint arXiv:2310.08582*, 2023.

[104] Ruida Hu, Chao Peng, Xinchen Wang, and Cuiyun Gao. An llm-based agent for reliable docker environment configuration. *arXiv preprint arXiv:2502.13681*, 2025.

[105] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

[106] Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023.

[107] Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Maarten Sap, and Michael R Lyu. On the resilience of multi-agent systems with malicious agents. *arXiv preprint arXiv:2408.00989*, 2024.

[108] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

[109] IBM. Introduction to agent communication protocol (acp). https://docs.beeai.dev/acp/alpha/introduction, 2024. Accessed: 2025.

[110] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

[111] P Vimala Imogen[1], J Sreenidhi, and V Nivedha. Ai-powered legal documentation assistant. *Journal of Artificial Intelligence*, 6(2):210–226, 2024.

[112] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[113] Yoshitaka Inoue, Tianci Song, and Tianfan Fu. Drugagent: Explainable drug repurposing agent with large language model-based reasoning. *arXiv preprint arXiv:2408.13378*, 2024.

[114] Yoichi Ishibashi and Yoshimasa Nishimura. Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization. *arXiv preprint arXiv:2404.02183*, 2024.

[115] Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*, 2025.

[116] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

[117] Shankar Kumar Jeyakumar, Alaa Alameer Ahmad, and Adrian Garret Gabriel. Advancing agentic systems: Dynamic task decomposition, tool integration and evaluation using novel metrics and dataset. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

[118] Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. Rag-thief: Scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks. *arXiv preprint arXiv:2411.14110*, 2024.

[119] Changyue Jiang, Xudong Pan, and Min Yang. Think twice before you act: Enhancing agent behavioral safety with thought correction. *arXiv preprint arXiv:2505.11063*, 2025.

[120] Huihao Jing, Haoran Li, Wenbin Hu, Qi Hu, Heli Xu, Tianshu Chu, Peizhao Hu, and Yangqiu Song. Mcip: Protecting mcp safety via model contextual integrity protocol. *arXiv preprint arXiv:2505.14590*, 2025.

[121] Erik Jones. Scalable auditing for ai safety. 2025.

[122] Matthew Joslin, Neng Li, Shuang Hao, Minhui Xue, and Haojin Zhu. Measuring and analyzing search engine poisoning of linguistic collisions. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1311–1325. IEEE, 2019.

[123] Tianjie Ju, Yiting Wang, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, and Gongshen Liu. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *arXiv preprint arXiv:2407.07791*, 2024.

[124] Mintong Kang, Chejian Xu, and Bo Li. Advwave: Stealthy adversarial jailbreak attack against large audio-language models. *arXiv preprint arXiv:2412.08608*, 2024.

[125] Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320, 2025.

[126] Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12140–12147. IEEE, 2024.

[127] Rana Muhammad Shahroz Khan, Zhen Tan, Sukwon Yun, Charles Flemming, and Tianlong Chen. Agents under siege: Breaking pragmatic multi-agent llm systems with optimized prompt attacks. *arXiv preprint arXiv:2504.00218*, 2025.

[128] Arsham Gholamzadeh Khoee, Shuai Wang, Yinan Yu, Robert Feldt, and Dhasarathy Parthasarathy. Gatelens: A reasoning-enhanced llm agent for automotive software release analytics. *arXiv preprint arXiv:2503.21735*, 2025.

[129] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

[130] Juhee Kim, Woohyuk Choi, and Byoungyoung Lee. Prompt flow integrity to prevent privilege escalation in llm agents. *arXiv preprint arXiv:2503.15547*, 2025.

[131] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[132] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[133] Ronny Ko, Jiseong Jeong, Shuyuan Zheng, Chuan Xiao, Taewan Kim, Makoto Onizuka, and Wonyong Shin. Seven security challenges that must be solved in cross-domain multi-agent llm systems. *arXiv preprint arXiv:2505.23847*, 2025.

[134] Dezhang Kong, Xiang Chen, Chunming Wu, Yi Shen, Zhengyan Zhou, Qiumei Cheng, Xuan Liu, Mingliang Yang, Yubing Qiu, Dong Zhang, et al. Rdefender: A lightweight and robust defense against flow table overflow attacks in sdn. *IEEE Transactions on Information Forensics and Security*, 2024.

[135] Dezhang Kong, Yi Shen, Xiang Chen, Qiumei Cheng, Hongyan Liu, Dong Zhang, Xuan Liu, Shuangxi Chen, and Chunming Wu. Combination attacks and defenses on sdn topology discovery. *IEEE/ACM Transactions on Networking*, 31(2):904–919, 2022.

[136] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[137] Naveen Krishnan. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *arXiv preprint arXiv:2504.21030*, 2025.

[138] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, pages 1097–1105, 2012.

[139] Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. Overthinking: Slowdown attacks on reasoning llms. *arXiv preprint arXiv:2502.02542*, 2025.

[140] Shrinidhi Kumbhar, Venkatesh Mishra, Kevin Coutinho, Divij Handa, Ashif Iquebal, and Chitta Baral. Hypothesis generation for materials

discovery and design using goal-driven and constraint-guided llm agents. *arXiv preprint arXiv:2501.13299*, 2025.

[141] Invariant Labs. Introducing mcp-scan: Protecting mcp with invariant. https://invariantlabs.ai/blog/introducing-mcp-scan, 2025. Accessed: 2025.

[142] LangChain. Agent protocol. https://github.com/langchain-ai/agent-protocol, 2024. Accessed: 2025.

[143] LangChain. Tool calling. https://python.langchain.com/docs/concepts/tool_calling/, 2024. Accessed: 2025.

[144] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models? *Advances in Neural Information Processing Systems*, 37:87874–87907, 2024.

[145] Tran Duc Le, Thang Le-Dinh, and Sylvestre Uwizeyemungu. Search engine optimization poisoning: A cybersecurity threat analysis and mitigation strategies for small and medium-sized enterprises. *Technology in Society*, 76:102470, 2024.

[146] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[147] Donghyun Lee and Mo Tiwari. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283*, 2024.

[148] Jingoo Lee, Kyungho Lim, Young-Chul Jung, and Byung-Hoon Kim. Psyche: A multi-faceted patient simulation framework for evaluation of psychiatric assessment conversational agents. *arXiv preprint arXiv:2501.01594*, 2025.

[149] Ash Lei. Atlassian tools & google a2a protocol: Revolutionizing ai agent interoperability. https://www.byteplus.com/en/topic/551110?title=atlassian-tools-google-a2a-protocol-revolutionizing-ai-agent-interoperability, 2025. Accessed: 2025.

[150] Bin Lei, Yi Zhang, Shan Zuo, Ali Payani, and Caiwen Ding. Macm: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. *arXiv preprint arXiv:2404.04735*, 2024.

[151] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[152] Ang Li, Yin Zhou, Vethavikashini Chithrra Raghuram, Tom Goldstein, and Micah Goldblum. Commercial llm agents are already vulnerable to simple yet dangerous attacks. *arXiv preprint arXiv:2502.08586*, 2025.

[153] Gaotang Li, Ting-Wei Li, and Xuying Ning. Mind the agent: A comprehensive survey on large language model-based agent safety. 2025.

[154] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*, 2023.

[155] Nian Li, Chen Gao, Mingyu Li, Yong Li, and Qingmin Liao. Econagent: large language model-empowered agents for simulating macroeconomic activities. *arXiv preprint arXiv:2310.10436*, 2023.

[156] Qiaomu Li and Ying Xie. From glue-code to protocols: A critical analysis of a2a and mcp integration for scalable agent systems. *arXiv preprint arXiv:2505.03864*, 2025.

[157] Rongchang Li, Minjie Chen, Chang Hu, Han Chen, Wenpeng Xing, and Meng Han. Gentel-safe: A unified benchmark and shielding framework for defending against prompt injection attacks. *arXiv preprint arXiv:2409.19521*, 2024.

[158] Xinzhe Li. A review of prominent paradigms for llm-based agents: Tool use (including rag), planning, and feedback learning. *arXiv preprint arXiv:2406.05804*, 2024.

[159] Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. Deepinception: Hypnotize large language model to be jailbreaker. In *Neurips Safe Generative AI Workshop 2024*, 2024.

[160] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

[161] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

[162] Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*, 2023.

[163] Shi Lin, Rongchang Li, Xun Wang, Changting Lin, Wenpeng Xing, and Meng Han. Figure it out: Analyzing-based jailbreak attack on large language models. *arXiv preprint arXiv:2407.16205*, 2024.

[164] Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 365–374, 2024.

[165] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[166] Haoyang Liu, Yijiang Li, Jinglin Jian, Yuxuan Cheng, Jianrong Lu, Shuyi Guo, Jinglei Zhu, Mianchen Zhang, Miantong Zhang, and Haohan Wang. Toward a team of ai-made scientists for scientific discovery from gene expression data. *arXiv preprint arXiv:2402.12391*, 2024.

[167] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

[168] Tong Liu, Yingjie Zhang, Zhe Zhao, Yinpeng Dong, Guozhu Meng, and Kai Chen. Making them ask and answer: Jailbreaking large language models in few queries via disguise and reconstruction. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4711–4728, 2024.

[169] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

[170] Xiaogeng Liu, Zhiyuan Yu, Yizhe Zhang, Ning Zhang, and Chaowei Xiao. Automatic and universal prompt injection attacks against large language models. *arXiv preprint arXiv:2403.04957*, 2024.

[171] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, et al. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023.

[172] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023.

[173] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1831–1847, 2024.

[174] Ziwen Liu, Jian Mao, Jun Zeng, Jiawei Li, Qixiao Lin, Jiahao Liu, Jianwei Zhuge, and Zhenkai Liang. Provguard: Detecting sdn control policy manipulation via contextual semantics of provenance graphs. In *NDSS*, 2025.

[175] Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024.

[176] Yuxuan Lu, Bingsheng Yao, Hansu Gu, Jing Huang, Jessie Wang, Yang Li, Jiri Gesi, Qi He, Toby Jia-Jun Li, and Dakuo Wang. Uxagent: An llm agent-based usability testing framework for web design. *arXiv preprint arXiv:2502.12561*, 2025.

[177] Huijie Lv, Xiao Wang, Yuansen Zhang, Caishuang Huang, Shihan Dou, Junjie Ye, Tao Gui, Qi Zhang, and Xuanjing Huang. Codechameleon: Personalized encryption framework for jailbreaking large language models. *arXiv preprint arXiv:2402.16717*, 2024.

[178] Siyuan Ma, Weidi Luo, Yu Wang, and Xiaogeng Liu. Visual-roleplay: Universal jailbreak attack on multimodal large language models via role-playing image character. *arXiv preprint arXiv:2405.20773*, 2024.

[179] Yingzi Ma, Yulong Cao, Jiachen Sun, Marco Pavone, and Chaowei Xiao. Dolphins: Multimodal language model for driving. In *European Conference on Computer Vision*, pages 403–420. Springer, 2024.

[180] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[181] Subhankar Maity and Aniket Deroy. Generative ai and its impact on personalized intelligent tutoring systems. *arXiv preprint arXiv:2410.10650*, 2024.

[182] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299. IEEE, 2024.

[183] Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. A language agent for autonomous driving. *arXiv preprint arXiv:2311.10813*, 2023.

[184] Junyuan Mao, Fanci Meng, Yifan Duan, Miao Yu, Xiaojun Jia, Junfeng Fang, Yuxuan Liang, Kun Wang, and Qingsong Wen. Agentsafe: Safeguarding large language model-based multi-agent systems via hierarchical data management. *arXiv preprint arXiv:2503.04392*, 2025.

[185] Ayman Maqsood, Chen Chen, and T Jesper Jacobsson. The future of material scientists in an age of artificial intelligence. *Advanced Science*, 11(19):2401401, 2024.

[186] Samuele Marro, Emanuele La Malfa, Jesse Wright, Guohao Li, Nigel Shadbolt, Michael Wooldridge, and Philip Torr. A scalable communication protocol for networks of large language models. *arXiv preprint arXiv:2410.11905*, 2024.

[187] Lauren Martin, Nick Whitehouse, Stephanie Yiu, Lizzie Catterson, and Rivindu Perera. Better call gpt, comparing large language models against lawyers. *arXiv preprint arXiv:2401.16212*, 2024.

[188] Evan Mattson. Integrating semantic kernel python with google's a2a protocol. https://devblogs.microsoft.com/foundry/semantic-kernel-a2a-integration/, 2025. Accessed: 2025.

[189] Sudhanshu Maurya, Arnav Kotiyal, Pankaj Prusty, Nilesh Shelke, Abhishek Bhattacherjee, and Tiyas Sarkar. Optimizing npc behavior in video games using unity ml-agents: A reinforcement learning-based approach. In *2025 3rd International Conference on Disruptive Technologies (ICDT)*, pages 1601–1606. IEEE, 2025.

[190] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

[191] Bertalan Meskó. Prompt engineering as an important emerging skill for medical professionals: tutorial. *Journal of medical Internet research*, 25:e50638, 2023.

[192] Erik Miehling, Karthikeyan Natesan Ramamurthy, Kush R Varshney, Matthew Riemer, Djallel Bouneffouf, John T Richards, Amit Dhurandhar, Elizabeth M Daly, Michael Hind, Prasanna Sattigeri, et al. Agentic ai needs a systems theory. *arXiv preprint arXiv:2503.00237*, 2025.

[193] Jakob Mökander, Jonas Schuett, Hannah Rose Kirk, and Luciano Floridi. Auditing large language models: a three-layered approach. *AI and Ethics*, 4(4):1085–1115, 2024.

[194] Moritz Möller, Gargi Nirmal, Dario Fabietti, Quintus Stierstorfer, Mark Zakhvatkin, Holger Sommerfeld, and Sven Schütt. Revolutionising distance learning: A comparative study of learning progress with ai-driven tutoring. *arXiv preprint arXiv:2403.14642*, 2024.

[195] Vineeth Sai Narajala, Ken Huang, and Idan Habler. Securing genai multi-agent systems against tool squatting: A zero trust registry-based approach. *arXiv preprint arXiv:2504.19951*, 2025.

[196] Imran Nasim. Governance in agentic workflows: Leveraging llms as oversight agents. In *AAAI 2025 Workshop on AI Governance: Alignment, Morality, and Law*, 2025.

[197] Fatemeh Nazary, Yashar Deldjoo, and Tommaso di Noia. Poison-rag: Adversarial data poisoning attacks on retrieval-augmented generation in recommender systems. In *European Conference on Information Retrieval*, pages 239–251. Springer, 2025.

[198] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.

[199] Yuzhou Nie, Zhun Wang, Ye Yu, Xian Wu, Xuandong Zhao, Wenbo Guo, and Dawn Song. Privagent: Agentic-based red-teaming for llm privacy leakage. *arXiv preprint arXiv:2412.05734*, 2024.

[200] Nvidia. Large language models explained. https://www.nvidia.com/en-us/glossary/large-language-models/, 2023. Accessed: 2025.

[201] Izunna Okpala, Ashkan Golgoon, and Arjun Ravi Kannan. Agentic ai systems applied to tasks in financial services: Modeling and model risk management crews. *arXiv preprint arXiv:2502.05439*, 2025.

[202] OpenAI. Function calling and other api updates. https://openai.com/index/function-calling-and-other-api-updates/, 2023. Accessed: 2025.

[203] OpenAI. Openai agent sdk supports mcp. https://openai.github.io/openai-agents-python/mcp/, 2025. Accessed: 2025.

[204] Xinming Ou, Sudhakar Govindavajhala, Andrew W Appel, et al. Mulval: A logic-based network security analyzer. In *USENIX security symposium*, volume 8, pages 113–128. Baltimore, MD, 2005.

[205] Joshua Owotogbe. Assessing and enhancing the robustness of llm-based multi-agent systems through chaos engineering. *arXiv preprint arXiv:2505.03096*, 2025.

[206] Paul Pajo. Ag-ui: Enabling user-facing ai agents through a lightweight event-based protocol for backend-frontend integration. 2025.

[207] Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, et al. Why do multiagent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025.

[208] Ioannis Papadimitriou, Ilias Gialampoukidis, Stefanos Vrochidis, and Ioannis Kompatsiaris. Ai methods in materials design, discovery and manufacturing: A review. *Computational Materials Science*, 235:112793, 2024.

[209] Keqin Peng, Liang Ding, Qihuang Zhong, Li Shen, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. Towards making the most of chatgpt for machine translation. *arXiv preprint arXiv:2303.13780*, 2023.

[210] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

[211] Samuele Poppi, Tobia Poppi, Federico Cocchi, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Safe-clip: Removing nsfw concepts from vision-and-language models. In *European Conference on Computer Vision*, pages 340–356. Springer, 2024.

[212] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*, 2023.

[213] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*, pages 5231–5240. PMLR, 2019.

[214] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[215] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[216] Lokman Rahmani, David Minarsch, and Jonathan Ward. Peer-to-peer autonomous agent communication network. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1037–1045, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[217] Rajesh Ranjan, Shailja Gupta, and Surya Narayan Singh. Loka protocol: A decentralized framework for trustworthy and ethical ai agent ecosystems. *arXiv preprint arXiv:2504.10915*, 2025.

[218] Surapaneni Rao, Jha Miku, Vakoc Michael, and Segal Todd. A new era of agent interoperability. https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/, 2025. Accessed: 2025.

[219] Charvi Rastogi, Marco Tulio Ribeiro, Nicholas King, Harsha Nori, and Saleema Amershi. Supporting human-ai collaboration in auditing llms with llms. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pages 913–926, 2023.

[220] Partha Pratim Ray. A review on agent-to-agent protocol: Concept, state-of-the-art, challenges and future directions. *Authorea Preprints*, 2025.

[221] Matthew Renze and Erhan Guven. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*, 2024.

[222] Grand View Research. Ai agents market size & trends. https://www.grandviewresearch.com/industry-analysis/ai-agents-market-report, 2024. Accessed: 2025.

[223] riseandignite. mcp-shield. https://github.com/riseandignite/mcp-shield, 2025. Accessed: 2025.

[224] Jaechul Roh, Virat Shejwalkar, and Amir Houmansadr. Multilingual and multi-accent jailbreaking of audio llms. *arXiv preprint arXiv:2504.01094*, 2025.

[225] Daniel Rose, Chia-Chien Hung, Marco Lepri, Israa Alqassem, Kiril Gashteovski, and Carolin Lawrence. Meddxagent: A unified modular agent framework for explainable automatic differential diagnosis. *arXiv preprint arXiv:2502.19175*, 2025.

[226] Yixiang Ruan, Chenyin Lu, Ning Xu, Yuchen He, Yixin Chen, Jian Zhang, Jun Xuan, Jianzhang Pan, Qun Fang, Hanyu Gao, et al. An automatic end-to-end chemical synthesis development platform powered by large language models. *Nature communications*, 15(1):10160, 2024.

[227] Gabriel Sarch, Lawrence Jang, Michael Tarr, William W Cohen, Kenneth Marino, and Katerina Fragkiadaki. Vlm agents generate their own memories: Distilling experience into embodied programs of thought. *Advances in Neural Information Processing Systems*, 37:75942–75985, 2024.

[228] Anjana Sarkar and Soumyendu Sarkar. Survey of llm agent communication with mcp: A software design pattern centric review. *arXiv preprint arXiv:2506.05364*, 2025.

[229] Robert Schwentker. Engineering the agentic future: Paypal's mcp & a2a design patterns for the new digital economy. https://www.linkedin.com/pulse/engineering-agentic-future-paypals-mcp-a2a-design-new-schwentker-w7hmc, 2025. Accessed: 2025.

[230] Shraddha Pradipbhai Shah and Aditya Vilas Deshpande. Enforcing cybersecurity constraints for llm-driven robot agents for online transactions. *arXiv preprint arXiv:2503.15546*, 2025.

[231] Gao Shang, Peng Zhe, Xiao Bin, Hu Aiqun, and Ren Kui. Floodde-fender: Protecting data and control plane resources under sdn-aimed dos attacks. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[232] Aarav Sharma, Kavya Gupta, and Rohan Jain. Building a secure agentic ai application leveraging google's a2a protocol. *arXiv preprint arXiv:2504.16902*, 2024.

[233] Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. *arXiv preprint arXiv:2307.14539*, 2023.

[234] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1671–1685, 2024.

[235] Xinyue Shen, Yixin Wu, Michael Backes, and Yang Zhang. Voice jailbreak attacks against gpt-4o. *arXiv preprint arXiv:2405.19103*, 2024.

[236] Jiawen Shi, Zenghui Yuan, Yinuo Liu, Yue Huang, Pan Zhou, Lichao Sun, and Neil Zhenqiang Gong. Optimization-based prompt injection attack to llm-as-a-judge. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 660–674, 2024.

[237] Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in llm agents. *arXiv preprint arXiv:2504.19793*, 2025.

[238] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[239] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. A survey of the model context protocol (mcp): Standardizing context to enhance large language models (llms). 2025.

[240] Shivam Singh, Karthik Swaminathan, Nabanita Dash, Ramandeep Singh, Snehasis Banerjee, Mohan Sridharan, and Madhava Krishna. Adaptbot: Combining llm with knowledge graphs and human input for generic-to-specific task decomposition and knowledge refinement. *arXiv preprint arXiv:2502.02067*, 2025.

[241] Li Song. Llm-driven npcs: Cross-platform dialogue system for games and social platforms. *arXiv preprint arXiv:2504.13928*, 2025.

[242] Ashwin Srinivasan, Karan Bania, Harshvardhan Mestha, Sidong Liu, et al. Implementation and application of an intelligibility protocol for interaction with an llm. *arXiv preprint arXiv:2410.20600*, 2024.

[243] V Stafford. Zero trust architecture. *NIST special publication*, 800(207):800–207, 2020.

[244] Shunqiao Sun, Athina P Petropulu, and H Vincent Poor. Mimo radar for advanced driver-assistance systems and autonomous driving: Advantages and challenges. *IEEE Signal Processing Magazine*, 37(4):98–117, 2020.

[245] Khritish Swargiary. The impact of ai-driven personalized learning and intelligent tutoring systems on student engagement and academic achievement: Ethical implications and the digital divide. *Available at SSRN 4897241*, 2024.

[246] Georgios Syros, Anshuman Suri, Cristina Nita-Rotaru, and Alina Oprea. Saga: A security architecture for governing ai agentic systems. *arXiv preprint arXiv:2504.21034*, 2025.

[247] Sagar Tamang and Dibya Jyoti Bora. Enforcement agents: Enhancing accountability and resilience in multi-agent ai frameworks. *arXiv preprint arXiv:2504.04070*, 2025.

[248] Dan Tang, Yudong Yan, Chenjun Gao, Wei Liang, and Wenqiang Jin. Ltrft: Mitigate the low-rate data plane ddos attack with learning-to-rank enabled flow tables. *IEEE Transactions on Information Forensics and Security*, 18:3143–3157, 2023.

[249] ANP Team. Agent network protocol (anp). https://agent-network-protocol.com/. Accessed 2025.

[250] Solo.io Engineering Team. Deep dive: Mcp and a2a attack vectors for ai agents. https://www.solo.io/blog/deep-dive-mcp-and-a2a-attack-vectors-for-ai-agents, 2024. Accessed: 2025.

[251] Tencent. Tencent cloud cos mcp server. https://github.com/Tencent/cos-mcp/blob/master/README.en.md, 2025. Accessed: 2025.

[252] Elizaveta Tennant, Stephen Hailes, and Mirco Musolesi. Moral alignment for llm agents. *arXiv preprint arXiv:2410.01639*, 2024.

[253] Kalliopi Terzidou. Generative ai systems in legal practice offering quality legal services while upholding legal ethics. *International Journal of Law in Context*, pages 1–22, 2025.

[254] PG Thirumagal, M Maria Antony Raj, Sarmad Jaafar Naser, Naseer Ali Hussien, Jamal K Abbas, and S Vinayagam. Efficient contract analysis and management through ai-powered tool: Time savings and error reduction in legal document review. In *2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, pages 1–6. IEEE, 2024.

[255] Khe-Han Toh and Hong-Kuan Teo. Modular speaker architecture: A framework for sustaining responsibility and contextual integrity in multi-agent ai communication. *arXiv preprint arXiv:2506.01095*, 2025.

[256] SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 6, 2024.

[257] Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. Can large language models really improve by self-critiquing their own plans? *arXiv preprint arXiv:2310.08118*, 2023.

[258] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[259] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008.

[260] Bo Wang, Weiyi He, Pengfei He, Shenglai Zeng, Zhen Xiang, Yue Xing, and Jiliang Tang. Unveiling privacy risks in llm agent memory. *arXiv preprint arXiv:2502.13172*, 2025.

[261] Haoyu Wang, Christopher M Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable llm agents. *arXiv preprint arXiv:2503.18666*, 2025.

[262] Jian Wang, Yinpei Dai, Yichi Zhang, Ziqiao Ma, Wenjie Li, and Joyce Chai. Training turn-by-turn verifiers for dialogue tutoring agents: The curious case of llms as your coding tutors. *arXiv preprint arXiv:2502.13311*, 2025.

[263] Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, et al. A comprehensive survey in llm (-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585*, 2025.

[264] Liwen Wang, Wenxuan Wang, Shuai Wang, Zongjie Li, Zhenlan Ji, Zongyi Lyu, Daoyuan Wu, and Shing-Chi Cheung. Ip leakage attacks targeting llm-based multi-agent systems. *arXiv preprint arXiv:2505.12442*, 2025.

[265] Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*, 2023.

[266] Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 639:130193, 2025.

[267] Ruida Wang, Rui Pan, Yuxin Li, Jipeng Zhang, Yizhen Jia, Shizhe Diao, Renjie Pi, Junjie Hu, and Tong Zhang. Ma-lot: Multi-agent lean-based long chain-of-thought reasoning enhances formal theorem proving. *arXiv preprint arXiv:2503.03205*, 2025.

[268] Ruofan Wang, Xingjun Ma, Hanxu Zhou, Chuanjun Ji, Guangnan Ye, and Yu-Gang Jiang. White-box multimodal jailbreaks against large vision-language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6920–6928, 2024.

[269] Shan Wang, Fang Wang, Zhen Zhu, Jingxuan Wang, Tam Tran, and Zhao Du. Artificial intelligence in education: A systematic literature review. *Expert Systems with Applications*, 252:124167, 2024.

[270] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37, 2024.

[271] Xin Wang, Yifan Zhang, Xiaojing Zhang, Longhui Yu, Xinna Lin, Jindong Jiang, Bin Ma, and Kaicheng Yu. Patentagent: Intelligent agent for automated pharmaceutical patent analysis. *arXiv preprint arXiv:2410.21312*, 2024.

[272] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

[273] Yaoxiang Wang, Zhiyong Wu, Junfeng Yao, and Jinsong Su. Tdag: A multi-agent framework based on dynamic task decomposition and agent generation. *Neural Networks*, page 107200, 2025.

[274] Yihan Wang, Zhouxing Shi, Andrew Bai, and Cho-Jui Hsieh. Defending llms against jailbreaking attacks via backtranslation. *arXiv preprint arXiv:2402.16459*, 2024.

[275] Yu Wang, Xiaofei Zhou, Yichen Wang, Geyuan Zhang, and Tianxing He. Jailbreak large visual language models through multi-modal linkage. *arXiv preprint arXiv:2412.00473*, 2024.

[276] Yuntao Wang, Shaolong Guo, Yanghe Pan, Zhou Su, Fahao Chen, Tom H Luan, Peng Li, Jiawen Kang, and Dusit Niyato. Internet of agents: Fundamentals, applications, and challenges. *arXiv preprint arXiv:2505.07176*, 2025.

[277] Yuntao Wang, Yanghe Pan, Shaolong Guo, and Zhou Su. Security of internet of agents: Attacks and countermeasures. *arXiv preprint arXiv:2505.08807*, 2025.

[278] Yuntao Wang, Yanghe Pan, Zhou Su, Yi Deng, Quan Zhao, Linkang Du, Tom H Luan, Jiawen Kang, and Dusit Niyato. Large model based agents: State-of-the-art, cooperation paradigms, security and privacy, and future trends. *IEEE Communications Surveys & Tutorials*, 2025.

[279] Ziyue Wang, Junde Wu, Chang Han Low, and Yueming Jin. Medagent-pro: Towards multi-modal evidence-based medical diagnosis via reasoning agentic workflow. *arXiv preprint arXiv:2503.18968*, 2025.

[280] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[281] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[282] Yunqian Wen, Bo Liu, Ming Ding, Rong Xie, and Li Song. Identitydp: Differential private identification protection for face images. *Neurocomputing*, 501:197–211, 2022.

[283] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

[284] WildCardAI. agents.json specification. https://github.com/wild-card-ai/agents-json, 2025. Accessed: 2025.

[285] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010, 2024.

[286] Yanchen Wu, Gangxin Xu, and Zou Dongchen. [proposal] self-reflection like humans, editable-llm (e-llm) is all you need. In *Tsinghua University Course: Advanced Machine Learning*, 2024.

[287] Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. Certifiably robust rag against retrieval corruption. *arXiv preprint arXiv:2405.15556*, 2024.

[288] Zhen Xiang, Fengqing Jiang, Zidi Xiong, Bhaskar Ramasubramanian, Radha Poovendran, and Bo Li. Badchain: Backdoor chain-of-thought prompting for large language models, 2024.

[289] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, et al. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

[290] Yihang Xiao, Jinyi Liu, Yan Zheng, Xiaohan Xie, Jianye Hao, Mingzhi Li, Ruitao Wang, Fei Ni, Yuxiao Li, Jintian Luo, et al. Cellagent: An llm-driven multi-agent framework for automated single-cell data analysis. *arXiv preprint arXiv:2407.09811*, 2024.

[291] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[292] Renjie Xie, Jiahao Cao, Qi Li, Kun Sun, Guofei Gu, Mingwei Xu, and Yuan Yang. Disrupting the sdn control channel via shared links: Attacks and countermeasures. *IEEE/ACM Transactions on Networking*, 30(5):2158–2172, 2022.

[293] Wenbei Xie, Donglin Liu, Haoran Yan, Wenjie Wu, and Zongyang Liu. Mathlearner: A large language model agent framework for learning to solve mathematical problems. *arXiv preprint arXiv:2408.01779*, 2024.

[294] Qi Xin, Quyu Kong, Hongyi Ji, Yue Shen, Yuqi Liu, Yan Sun, Zhilin Zhang, Zhaorong Li, Xunlong Xia, Bing Deng, et al. Bioinformatics agent (bia): Unleashing the power of large language models to reshape bioinformatics workflow. *BioRxiv*, pages 2024–05, 2024.

[295] Lei Xu, Jeff Huang, Sungmin Hong, Jialong Zhang, and Guofei Gu. Attacking the brain: Races in the {SDN} control plane. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 451–468, 2017.

[296] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*, 2024.

[297] Jiaqi Xue, Mengxin Zheng, Yebowen Hu, Fei Liu, Xun Chen, and Qian Lou. Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models, 2024.

[298] Bingyu Yan, Xiaoming Zhang, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, and Chaozhuo Li. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *arXiv preprint arXiv:2502.14321*, 2025.

[299] Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, et al. A survey of ai agent protocols. *arXiv preprint arXiv:2504.16736*, 2025.

[300] Yuzhe Yang, Yifei Zhang, Minghao Wu, Kaidi Zhang, Yunmiao Zhang, Honghai Yu, Yan Hu, and Benyou Wang. Twinmarket: A scalable behavioral and socialsimulation for financial markets. *arXiv preprint arXiv:2502.01506*, 2025.

[301] Zuopeng Yang, Jiluan Fan, Anli Yan, Erdun Gao, Xin Lin, Tao Li, Kanghua Mo, and Changyu Dong. Distraction is all you need for multimodal large language model jailbreaking. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9467–9476, 2025.

[302] Shanliang Yao, Runwei Guan, Zitian Peng, Chenhang Xu, Yilu Shi, Weiping Ding, Eng Gee Lim, Yong Yue, Hyungjoon Seo, Ka Lok Man, et al. Exploring radar data representations in autonomous driving: A comprehensive review. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[303] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

[304] Zonghao Ying, Aishan Liu, Tianyuan Zhang, Zhengmin Yu, Siyuan Liang, Xianglong Liu, and Dacheng Tao. Jailbreak vision language models via bi-modal adversarial prompt. *arXiv preprint arXiv:2406.04031*, 2024.

[305] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. LLM-Fuzzer: Scaling assessment of large language model jailbreaks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4657–4674, Philadelphia, PA, August 2024. USENIX Association.

[306] Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pang, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, et al. A survey on trustworthy llm agents: Threats and countermeasures. *arXiv preprint arXiv:2503.09648*, 2025.

[307] Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Qingsong Wen, Kun Wang, and Yang Wang. Netsafe: Exploring the topological safety of multi-agent networks. *arXiv preprint arXiv:2410.15686*, 2024.

[308] Peiying Yu, Guoxin Chen, and Jingjing Wang. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. *arXiv preprint arXiv:2502.11799*, 2025.

[309] Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design. In *Proceedings of the AAAI Symposium Series*, volume 3, pages 595–597, 2024.

[310] Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan Suchow, Zhenyu Cui, Rong Liu, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045, 2024.

[311] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*, pages 841–852, 2022.

[312] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201*, 2024.

[313] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, and Jiliang Tang. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag), 2024.

[314] Wenjun Zeng et al. Shieldgemma 2: Robust and tractable image content moderation. 2025.

[315] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193*, 2024.

[316] Boyang Zhang, Yicong Tan, Yun Shen, Ahmed Salem, Michael Backes, Savvas Zannettou, and Yang Zhang. Breaking agents: Compromising autonomous llm agents through malfunction amplification. *arXiv preprint arXiv:2407.20859*, 2024.

[317] Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*, 2025.

[318] Jieyu Zhang, Ranjay Krishna, Ahmed H Awadallah, and Chi Wang. Ecoassistant: Using llm assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046*, 2023.

[319] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[320] Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025.

[321] Kaiyuan Zhang, Zian Su, Pin-Yu Chen, Elisa Bertino, Xiangyu Zhang, and Ninghui Li. Llm agents should employ security principles. *arXiv preprint arXiv:2505.24019*, 2025.

[322] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

[323] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5579–5588, 2021.

[324] Qingzhao Zhang, Ziyang Xiong, and Z Morley Mao. Safeguard is a double-edged sword: Denial-of-service attack on large language models. *arXiv preprint arXiv:2410.02916*, 2024.

[325] Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023.

[326] Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards efficient llm grounding for embodied multi-agent collaboration. *arXiv preprint arXiv:2405.14314*, 2024.

[327] Yiqun Zhang, Xiaocui Yang, Xiaobai Li, Siyuan Yu, Yi Luan, Shi Feng, Daling Wang, and Yifei Zhang. Psydraw: A multi-agent multimodal system for mental health screening in left-behind children. *arXiv preprint arXiv:2412.14769*, 2024.

[328] Yuanhe Zhang, Zhenhong Zhou, Wei Zhang, Xinyue Wang, Xiaojun Jia, Yang Liu, and Sen Su. Crabs: Consuming resrouce via auto-generation for llm-dos attack under black-box settings. *arXiv preprint arXiv:2412.13879*, 2024.

[329] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

[330] Yuqi Zhang, Liang Ding, Lefei Zhang, and Dacheng Tao. Intention analysis makes llms a good jailbreak defender. *arXiv preprint arXiv:2401.06561*, 2024.

[331] Zaibin Zhang, Yongting Zhang, Lijun Li, Hongzhi Gao, Lijun Wang, Huchuan Lu, Feng Zhao, Yu Qiao, and Jing Shao. Psysafe: A comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety. *arXiv preprint arXiv:2401.11880*, 2024.

[332] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. 2024.

[333] Zhexin Zhang, Yida Lu, Jingyuan Ma, Di Zhang, Rui Li, Pei Ke, Hao Sun, Lei Sha, Zhifang Sui, Hongning Wang, et al. Shieldlm: Empowering llms as aligned, customizable and explainable safety detectors. *arXiv preprint arXiv:2402.16444*, 2024.

[334] Zhiping Zhang, Bingcan Guo, and Tianshi Li. Privacy leakage overshadowed by views of ai: A study on human oversight of privacy in language model agent. *arXiv preprint arXiv:2411.01344*, 2024.

[335] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.

[336] Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36:31967–31987, 2023.

[337] Can Zheng, Yuhan Cao, Xiaoning Dong, and Tianxing He. Demonstrations of integrity attacks in multi-agent systems. *arXiv preprint arXiv:2506.04572*, 2025.

[338] Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L Titzer, Heather Miller, and Phillip B Gibbons. Rtbas: Defending llm agents against prompt injection and privacy leakage. *arXiv preprint arXiv:2502.08966*, 2025.

[339] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198*, 2023.

[340] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.

[341] Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. Poisoning retrieval corpora by injecting adversarial passages, 2023.

[342] Huichi Zhou, Kin-Hei Lee, Zhonghao Zhan, Yue Chen, and Zhenhao Li. Trustrag: Enhancing robustness and trustworthiness in rag. *arXiv preprint arXiv:2501.00879*, 2025.

[343] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[344] Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE, 2024.

[345] Zhenhong Zhou, Zherui Li, Jie Zhang, Yuanhe Zhang, Kun Wang, Yang Liu, and Qing Guo. Corba: Contagious recursive blocking attacks on multi-agent systems based on large language models. *arXiv preprint arXiv:2502.14529*, 2025.

[346] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[347] Ren Zhuang, Ben Wang, and Shuifa Sun. Accelerating chain-of-thought reasoning: When goal-gradient importance meets dynamic skipping. *arXiv preprint arXiv:2505.08392*, 2025.

[348] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*, 2024.