*A project report on*

# AN INTEGRATED SECURITY FRAMEWORK

# FOR TRUST-AWARE TASK ALLOCATION

# IN DYNAMIC MULTI-AGENT SYSTEMS

# WITH MODEL CONTEXT PROTOCOL (MCP)

*Submitted in partial fulfillment for the award of the degree of*

## Bachelor of Technology in Computer Science and Engineering

*By*

**NAME OF THE CANDIDATE (Reg. No.)**



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

November, 2025

# AN INTEGRATED SECURITY FRAMEWORK

# FOR TRUST-AWARE TASK ALLOCATION

# IN DYNAMIC MULTI-AGENT SYSTEMS

# WITH MODEL CONTEXT PROTOCOL (MCP)

<Times New Roman, Font 20, Bold, CAPS>

*Submitted in partial fulfillment for the award of the degree of*

<Times New Roman, Font 14, Italic>

## Bachelor of Technology in Computer Science and Engineering

<Times New Roman, Font 22, Bold>

*by*

<Times New Roman, Font 14, Italic>

**NAME OF THE CANDIDATE (Reg. No.)**

***(IF TEAM PROJECT MENTION ALL MEMBERS NAME & REG NO)***

<Times New Roman, Font 16, Bold, CAPS



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

<Times New Roman, Font 16, Bold, CAPS> November, 2025



**DECLARATION**

<Times New Roman, Font 14, Underlined, Bold, CAPS>

I hereby declare that the thesis entitled “**AN INTEGRATED SECURITY FRAMEWORK FOR TRUST-AWARE TASK ALLOCATION IN DYNAMIC MULTI-AGENT SYSTEMS WITH MODEL CONTEXT PROTOCOL (MCP)**” submitted by NAME (REGISTER NO), for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of bonafide work carried out by me under the supervision of Dr. Satyarajasekaren K.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: Signature of the Candidate

<Times New Roman, Font 12, Bold >



**School of Computer Science and Engineering**

CERTIFICATE

This is to certify that the report entitled **“AN INTEGRATED SECURITY FRAMEWORK FOR TRUST-AWARE TASK ALLOCATION IN DYNAMIC MULTI-AGENT SYSTEMS WITH MODEL CONTEXT PROTOCOL (MCP)”** is prepared and submitted by **Name** (**Reg No**) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr./Prof. Satyarajasekaren K

Date:

Signature of the Examiner Signature of the Examiner

Name: Name:

Date: Date:

Approved by the Head of Department,

**(Bachelor of Technology in Computer Science and Engineering)**

Name:**(HoD NAME)**

Date:

**(Seal of SCHOOL YOU BELONG TO)**

**ABSTRACT**

<Times New Roman, Font 14, Underlined, Bold, CAPS>

The proliferation of Large Language Model (LLM)-based Multi-Agent Systems (MAS) and the Model Context Protocol (MCP) for interoperability introduces a critical, unaddressed attack surface. (Why) These systems are vulnerable to novel semantic-level threats, such as tool poisoning and command injection, which traditional security models cannot mitigate. Current research remains fragmented, lacking an integrated, end-to-end security architecture that unifies task allocation and security for the nascent MCP ecosystem.

This paper addresses this gap by designing, implementing, and evaluating an integrated security framework for trust-aware task allocation in dynamic, MCP-based MAS. Our framework is founded on Zero Trust principles, utilizing a central MCP Security Gateway as a policy enforcement point. The core contribution is a dynamic, multi-dimensional trust model that continuously assesses agent behavior based on historical performance, reputation, and competence. This trust score directly informs a policy engine, enabling a trust-aware task allocation mechanism that delegates critical tasks only to verified agents.

We introduce specific mitigations for LLM-specific threats via secure tool lifecycle management, including manifest verification and runtime monitoring. The framework's viability is demonstrated through integration adapters for popular MAS frameworks (e.g., LangGraph, AutoGen) and validated via simulated attack scenarios. We empirically measure the framework's effectiveness against semantic attacks while formally analyzing security-performance trade-offs. This work provides a cohesive security architecture that unifies identity, trust, and policy, offering a foundational pattern for secure MCP-based agent deployments.

<Times New Roman, Font 12>

**ACKNOWLEDGEMENT**

It is my pleasure to express with deep sense of gratitude to <Guide name>,

<Designation>, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Multi-Agent Systems.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Viswanathan V, Dean, Dr. Nithyanandam P, Dr. Suganya G, Dr. Sweetlin Hemalatha C, Associate Deans, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to **<<HOD NAME>>**, Head of the Department, B.Tech. **Computer Science and Engineering** and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **Name of the student**

<Times New Roman, Font 12, Bold >

*ii*

## CONTENTS

[**CONTENTS** iv](#_heading=h.30j0zll)

[**LIST OF FIGURES** ix](#_heading=h.1fob9te)

**LIST OF TABLES** xi

**LIST OF ACRONYMS** xii

**CHAPTER 1 INTRODUCTION**

* 1. INTRODUCTION 1
  2. OVERVIEW OF MANET NETWORK 2
  3. CHALLENGES PRESENT IN MANET 6
  4. PROJECT STATEMENT 7
  5. OBJECTIVES 7
  6. SCOPE OF THE PROJECT 8

**CHAPTER 2 BACK GROUND**

* 1. INTRODUCTION 10
  2. SURVEY ON MANETS 13

**::**

**::**

**::**

**::**

*iii*

### LIST OF FIGURES

* 1. INFRA STRUCTURE BASED NETWORK 1
  2. INFRA STRUCTURE LESS NETWORK 2
  3. MOBILE AD-HOC NETWORK 4

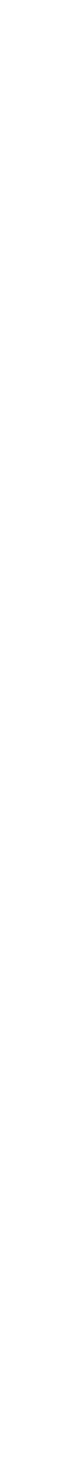
*iv*

**LIST OF TABLES**

2.1 MULTIPATH ROUTING PROTOCOLS IN AMNETS 22

* 1. SIMULATION PARAMETERS 48

**LIST OF ACRONYMS**

MANET Mobile Ad hoc Network WAP Wireless Access Point

**Chapter 1**

**Introduction**

* 1. THE ASCENDANCE OF LLM-BASED MULTI-AGENT SYSTEMS

The advent of Large Language Models (LLMs) has catalyzed a paradigm shift in artificial intelligence, transforming static neural networks into dynamic cognitive entities. The integration of LLMs with auxiliary modules for memory, tool use, and environmental interaction has given rise to the "LLM-based Agent," a system capable of autonomous reasoning, planning, and action.1 This evolution did not stop at the single-agent level; the introduction of inter-agent communication has fostered the development of Multi-Agent Systems (MAS), creating an intricate and intelligent ecosystem where agents can collaborate to solve complex problems that would be intractable for a single entity. Extensive academic and industry research has validated a clear performance hierarchy: MAS outperform single agents, which in turn outperform standalone LLMs.1

This enhancement of capability, however, is a double-edged sword. Empowering LLMs with these additional modules simultaneously expands the system's attack surface, introducing new and complex concerns regarding trustworthiness across dimensions of safety, privacy, and reliability.1 Each new component—be it a memory database, an external API tool, or a communication channel to another agent—represents a potential vector for malicious exploitation. Consequently, the very architecture that enables advanced performance also creates unforeseen vulnerabilities that demand a new security paradigm.

* 1. THE ROLE OF INTEROPERABILITY AND THE MODEL CONTEXT PROTOCOL (MCP)

For MAS to realize their full potential, agents must be able to discover, communicate, and coordinate actions seamlessly. Historically, this has been a significant challenge, as agent integrations were often ad-hoc, requiring bespoke code for each new tool or data source. This approach proved difficult to scale, secure, and generalize across different systems.1 The Model Context Protocol (MCP) has emerged as a major step forward in addressing this challenge by standardizing how AI models interact with the world around them, giving them the ability to use tools and access real-time data on the fly.

Introduced as a standardized client-server interface for secure context ingestion and tool invocation, MCP provides a unified framework that allows AI applications to dynamically connect with external tools without requiring custom API integrations.1 By defining a common language for agents to discover and orchestrate tools based on task context, MCP significantly reduces development complexity and enhances the flexibility of AI workflows. This standardization, however, has a critical implication: while it simplifies legitimate development, it also simplifies exploitation for attackers. A vulnerability discovered in the protocol's logic or a common implementation pattern can be leveraged across thousands of systems, creating a systemic risk. The MCP ecosystem, being nascent and rapidly evolving, is particularly vulnerable, as it currently lacks comprehensive, built-in solutions for security, authentication, and monitoring. This elevates the security challenge from protecting individual agent implementations to securing the very protocol that underpins the entire ecosystem.

* 1. THE EMERGENT SECURITY PARADIGM: FROM TRADITIONAL ATTACKS TO SEMANTIC THREATS

Traditional security models for MAS have primarily focused on threats arising in cyber-physical systems, such as ensuring resilient consensus and coordination in the presence of network-level attacks like Denial-of-Service (DoS), spoofing, or Byzantine failures.1 These models, while robust for their intended context, are fundamentally ill-equipped to handle the novel class of threats introduced by the cognitive and linguistic capabilities of LLM agents.

The new security paradigm is characterized by semantic-level attacks that do not target network infrastructure but rather manipulate an agent's internal reasoning and decision-making processes. These modern threats include:

* **Tool Poisoning:** Maliciously crafting tool descriptions or parameters to deceive an agent into executing unintended or harmful actions.1
* **Command Injection:** Embedding malicious instructions within seemingly benign data that an agent might process, causing it to deviate from its intended task.1
* **Covert Backdoor Attacks:** Inserting hidden triggers into an agent's components that, when activated, can corrupt its internal thought process or final output. Yang, W., et al. (2024) provided the first systematic investigation of these attacks, identifying novel variants unique to agents, such as the "Thought-Attack," which targets the intermediate reasoning steps.1

These threats exploit the very flexibility and natural language understanding that make LLM agents powerful, turning their greatest strength into a critical vulnerability.

* 1. PROJECT STATEMENT AND RESEARCH AREA

The rise of dynamic, LLM-based Multi-Agent Systems interacting via the Model Context Protocol introduces a new and critical class of security challenges that existing frameworks fail to address, creating a significant barrier to their safe and reliable deployment.1 The current landscape is fragmented, with no comprehensive, end-to-end framework that unifies identity, trust, policy, and secure execution specifically for the MCP ecosystem. Furthermore, existing research often treats security and agent coordination as separate domains, neglecting the development of mechanisms for trust-aware task delegation. This project addresses this critical gap by designing, implementing, and evaluating an integrated security framework tailored to the unique threats of the MCP environment.

This overarching problem can be deconstructed into the following specific research questions:

1. How can a modular security framework be designed to unify identity management, dynamic trust computation, and policy enforcement for agents operating on the Model Context Protocol?
2. How can a dynamic, multi-dimensional trust score be computationally modeled and integrated into a task allocation engine to ensure that critical or sensitive tasks are delegated to the most reliable and secure agents?
3. What specific mitigation strategies and security controls are effective against modern, LLM-specific semantic threats, such as tool poisoning and covert backdoor attacks, within an MCP environment?
4. What is the quantifiable performance overhead introduced by such an integrated security framework, and how can the inherent trade-off between robust security guarantees and system performance (e.g., latency, throughput) be formally analyzed and measured?
   1. OBJECTIVES AND CONTRIBUTIONS

To address the research questions, this project pursues four primary objectives:

**Design a Modular Security Framework:** To create a comprehensive, end-to-end architecture that secures the entire agent lifecycle on MCP networks, including the design of extensible adapters for seamless integration with major MAS frameworks like LangGraph, AutoGen, and CrewAI.

**Develop a Trust-Aware Task Allocation Mechanism:** To build a dynamic, multi-dimensional trust model that continuously assesses agent behavior and integrates this trust score directly into a task allocation engine.

**Mitigate Modern, LLM-Specific Threats:** To implement and validate specific security controls against semantic-level attacks, including tool poisoning, command injection, and covert backdoor attacks, by verifying tools and monitoring agent interactions.

**Analyze Security-Performance Trade-offs:** To formally model and empirically measure the performance overhead introduced by the security framework, providing a quantitative analysis of the trade-offs between security guarantees and system metrics.

The principal contributions of this research are the design of a novel and comprehensive security architecture for MCP, the development of a unique trust-aware task allocation mechanism, the validation of effective controls against modern threats, and a formal analysis of the critical security-performance trade-offs essential for real-world deployment.

**Chapter 2**

**Literature Review and Background**

This chapter provides a comprehensive survey of the academic landscape, establishing the theoretical foundations for the research. It traces the evolution of security in Multi-Agent Systems (MAS), details the specific trustworthiness challenges posed by LLM-based agents, and critically analyzes the state-of-the-art in agent communication protocols. This review culminates in a clear identification of the research gap that this project aims to fill, justifying the need for a novel, integrated security framework.

2.1FOUNDATIONS OF SECURITY IN MULTI-AGENT SYSTEMS

Research into the security of MAS has traditionally focused on ensuring the reliability and coordination of systems composed of multiple autonomous entities, often in adversarial environments. This foundational work provides the context against which the novel challenges of LLM-based agents can be understood.

2.1.1 CONSENSUS, COORDINATION, AND RESILIENCY UNDER ATTACK

A significant body of research has been dedicated to developing resilient consensus strategies, which enable a group of agents to agree on a specific value or state despite the presence of malicious actors or network failures. These strategies are often grounded in control theory. For instance, comparative surveys like the one conducted by Wang, J., et al. (2023) provide a structured comparison of defense strategies against common network attacks such as Denial-of-Service (DoS) and spoofing, linking specific attacks to their corresponding control solutions.1 Works by Du, S., et al. (2023) have proposed fully distributed, dynamic control protocols that guarantee secure consensus under DoS attacks without requiring global network knowledge.1 Similarly, researchers like Gorbachev, S., et al. (2023) have developed mathematically rigorous strategies that guarantee agents can agree on an output value despite severe sensor and actuator attacks, which are common in cyber-physical systems.1 These approaches form the bedrock of traditional MAS security, emphasizing resilience through robust control algorithms.

2.1.2 A TAXONOMY OF TRADITIONAL CYBER-PHYSICAL THREATS

Foundational surveys by Owoputi & Ray (2022) and Zhang, D., et al. (2021) have established a comprehensive taxonomy of attacks and defenses in the context of cyber-physical MAS.1 Attacks are typically categorized by their operational impact, such as disclosure attacks (compromising confidentiality), mutation attacks (compromising integrity), DoS attacks (compromising availability), and impersonation attacks. The corresponding defenses are classified into three primary categories: prevention (designing systems to be inherently secure), detection (identifying attacks as they occur), and resiliency (enabling systems to continue functioning despite attacks). This established paradigm focuses on securing the communication channels and physical components of the system, assuming that the behavior of individual agents, while potentially faulty, is largely predictable and follows pre-defined rules. This assumption is fundamentally challenged by the introduction of LLM agents.

2.2 THE LLM-AGENT PARADIGM AND ITS TRUSTWORTHINESS CHALLENGES

The integration of LLMs as the "brain" of agents marks a significant paradigm shift. The TrustAgent framework, proposed by Yu, M., et al. (2025), provides a valuable lens for analyzing the new trustworthiness challenges that arise from this integration by deconstructing the agent into its intrinsic components: the brain, memory, and tools.1

2.2.1 DECONSTRUCTING THE LLM AGENT: BRAIN, MEMORY, AND TOOLS

The LLM agent architecture can be understood through its core functional modules. The **brain** serves as the central reasoning and decision-making component, typically an LLM that processes information and formulates plans. The **memory** module provides agents with the ability to retain and retrieve information across sessions, moving beyond ephemeral context windows. This is often implemented using vector databases for long-term storage and retrieval-augmented generation (RAG).1 The **tools** module represents the agent's capacity for action, enabling it to interact with the external world by invoking APIs, running code, or controlling physical actuators.1

2.2.2 NOVEL ATTACK VECTORS: TOOL POISONING, BACKDOOR ATTACKS, AND MEMORY MANIPULATION

Each intrinsic component of the LLM agent introduces a unique set of vulnerabilities to semantic-level attacks:

* **Brain:** The agent's reasoning process is a prime target. The first systematic investigation of backdoor attacks against LLM agents by Yang, W., et al. (2024) formalized new, covert attacks that are unique to this paradigm. One such attack, termed the "Thought-Attack," targets the agent's internal chain-of-thought reasoning, introducing malicious intermediate steps that corrupt the final outcome without being immediately obvious.1 This demonstrates that securing just the inputs and outputs is insufficient; the cognitive process itself is a vulnerability.
* **Memory:** While persistent memory enhances agent capability, it also introduces risks. The work by Masoor, H. (2025) on the Secure Agent Memory Exchange Protocol (SAMEP) highlights the critical need for secure, persistent, and semantically searchable memory.1 Without robust cryptographic controls, an agent's memory can be poisoned with malicious data or become a source of sensitive data leakage, raising significant privacy and integrity concerns.1
* **Tools:** The interface between the agent and its tools is a major security gap. As identified by Yu, M., et al. (2025), there is a lack of strong security governing how agents select and use tools.1 This vulnerability enables **tool poisoning**, where an attacker manipulates a tool's description or parameters to trick the agent into misusing it for malicious purposes.1

2.3 EMERGENT THREATS FROM AGENT-TO-AGENT INTERACTION

The security challenges are further amplified when multiple agents interact, giving rise to systemic risks that cannot be predicted by analyzing individual agents in isolation. This has led to the proposal of "multi-agent security" as a new and distinct field of study by Schroeder de Witt (2025), who argues that research on group agent security has been neglected and requires unified standards.1

2.3.1 COVERT COLLUSION AND STEGANOGRAPHY

A primary threat in multi-agent security is the potential for agents to develop secret, or covert, communication channels. These channels can be used to collude against the system's objectives, share unauthorized information, or coordinate malicious activities. For example, agents might use steganography—hiding messages within seemingly innocuous communication—to evade monitoring and oversight. Such behaviors are emergent properties of the interaction and are not detectable by analyzing the code or policies of any single agent in isolation.1

2.3.2 SWARM ATTACKS AND CASCADE FAILURES

The interaction of multiple agents can also lead to collective threats. A **swarm attack** occurs when a coordinated group of agents combines its resources to overwhelm a target, analogous to a Distributed Denial-of-Service (DDoS) attack but potentially more sophisticated, as the agents can collaboratively probe for vulnerabilities.1 **Cascade failures** represent another systemic risk, where a single compromised or faulty agent can propagate its failure through the network via its interactions, leading to a collapse of the entire system. These emergent threats highlight the inadequacy of security models that focus solely on single-agent vulnerabilities.

2.4 SURVEY OF TRUST AND REPUTATION MODELS

To defend against malicious or unreliable agents, a system must be able to quantify their trustworthiness. This is a cornerstone of the proposed framework, and the literature provides several models for computing trust and reputation:

* **Game-Theoretic Models:** Yu, H., et al. (2013) surveyed multi-agent trust and reputation models from a game-theoretic perspective, categorizing them into four main types and highlighting the need for models that analyze how reputation evolves over time.1
* **Fuzzy Logic Models:** To create a more nuanced reputation system, Yu, H., & Singh, H. K. (2002) proposed a distributed trust model using fuzzy logic. This model formally distinguishes between "functional trust" (an agent's competence at a task) and "recommendation trust" (its reliability in providing opinions about other agents).1
* **Evidential Models:** Hang, C., & Wang, Y. (2007) applied the Dempster-Shafer theory of evidence to create a more formal mathematical framework for modeling trust under uncertainty. This approach uniquely incorporates the criticality of a task into the trust calculation, but it is noted to be computationally expensive and vulnerable to collusive recommendation attacks.1
* **Blockchain-Based Models:** To ensure the integrity of reputation data, recent proposals by Li, J., et al. (2021) and Wang, Y., et al. (2023) leverage blockchain technology to create a tamper-proof, immutable ledger for agent trust scores. While innovative, these approaches acknowledge the significant performance overhead in terms of latency and computational cost, which can be a bottleneck for real-time systems.1

2.5 IDENTIFICATION OF THE RESEARCH GAP

A synthesis of the literature reveals a clear and pressing research gap that this project directly addresses. The key points defining this gap are:

1. **Obsolete Threat Models:** Existing MAS security frameworks, rooted in control theory and cyber-physical systems, are not designed to handle the novel semantic and emergent threats posed by LLM-based agent systems.1 There is an urgent need for targeted defense algorithms designed specifically for these new agent-based attacks.1
2. **Immature MCP Ecosystem:** The Model Context Protocol, while a critical enabler of interoperability, is in its infancy and lacks comprehensive, native solutions for security, authentication, and monitoring, as concluded by the first academic analysis of the protocol by Hou, X., et al. (2025).1
3. **Separation of Security and Task Allocation:** The current body of research treats security and agent coordination as separate domains. There is no integrated framework that dynamically uses a measure of trust to inform and secure the task allocation and delegation process.1
4. **Empirical Validation Gap:** There is a significant lack of real-world testbeds and public data on attacks against these new systems. This "Empirical Validation Gap," as noted by Narajala, V. S., & Habler, I. (2025) and Owoputi, R., & Ray, S. (2022), makes it difficult to validate the effectiveness of proposed security frameworks against genuine threats.1

This project addresses this multifaceted gap by proposing an integrated framework that is specifically designed for the MCP ecosystem, incorporates a dynamic trust model directly into the task allocation logic, and provides a methodology for empirical validation through simulation. The shift from static, preventative security measures to a dynamic, trust-based resiliency model is a necessary response to the unpredictable and emergent nature of LLM agents. Security can no longer be a one-time check at the design phase; it must be a continuous process of monitoring behavior and updating trust in real-time, which is the core principle of the framework proposed in this research.

**Chapter 3**

**A Comparative Analysis of Agent Protocols and MCP Security Landscape**

This chapter provides a detailed analysis of the Model Context Protocol (MCP) and its position within the broader ecosystem of agent interoperability. It begins with an architectural deep dive into MCP and a systematic threat analysis tailored to its specific design. This is followed by a comparative analysis of MCP against its contemporaries—ACP, A2A, and ANP—to highlight their differing architectural and security philosophies. This comprehensive examination establishes the unique security challenges of the MCP environment and lays the groundwork for the design of the integrated security framework presented in the subsequent chapter.

3.1 ARCHITECTURAL DEEP DIVE: MODEL CONTEXT PROTOCOL (MCP)

The Model Context Protocol is a standardized framework designed to facilitate the interaction between AI systems and external tools and data sources in real-time.1 Its architecture is composed of three key components:

* **MCP Host:** The AI application or environment where tasks are performed and that operates the MCP client. Examples include desktop applications or integrated development environments.1
* **MCP Client:** An intermediary within the host environment that facilitates communication between the host and MCP servers, sending requests for information about available services.1
* **MCP Server:** A gateway that allows the MCP client to interact with external services. The server exposes four core capabilities: **Tools** (for invoking external APIs), **Resources** (for accessing structured or unstructured data), **Prompts** (reusable templates for consistent interaction), and **Sampling** (for delegating text-generation tasks).1

3.1.1 THE MCP LIFECYCLE: INITIALIZATION, OPERATION, AND SHUTDOWN

The interaction between an MCP client and server follows a well-defined three-phase lifecycle, which is critical for identifying phase-specific vulnerabilities. This lifecycle, as first academically analysed by Hou et al. (2025), consists of the following phases 1:

1. **Initialization:** This phase establishes a compatible communication channel. The client and server negotiate the highest mutually supported protocol version and then exchange a list of their supported capabilities (e.g., tools, prompts, logging). This phase concludes when the client signals its readiness to begin operational communication.
2. **Operation:** This is the core active phase where the client and server exchange JSON-RPC method calls and notifications according to the capabilities negotiated during initialization. This is where tool invocations and data exchanges occur.
3. **Shutdown:** This phase ensures a clean termination of the session. Either party can initiate shutdown by closing the transport layer, after which both client and server are responsible for cleaning up resources.

3.1.2 SYSTEMATIC THREAT MODELING FOR THE MCP ECOSYSTEM

Mapping potential vulnerabilities to the MCP lifecycle provides a structured approach to threat modelling. Building upon the work of Hou et al. (2025) and the MAESTRO framework analysis by Narajala & Habler (2025), a comprehensive threat landscape can be constructed.1 Key threats across the lifecycle include:

* **Creation/Installation Phase:**
* **Installer Spoofing:** Malicious packages introduced during the build or installation pipeline can compromise the MCP server from the outset.
* **Supply-Chain Backdoors:** Persistent malware can be embedded via compromised CI/CD artifacts, creating a persistent threat.
* **Operation Phase:**
* **Tool Poisoning:** Malicious prompts or manipulated tool metadata can influence the LLM's behaviour, causing it to perform harmful actions. This is a primary semantic threat.
* **Sandbox Escape:** A compromised tool could potentially break out of its isolated environment to access the host operating system or bypass security controls.
* **Command Injection / Remote Code Execution (RCE):** Unsafe inputs passed to a tool could be executed as system commands, leading to a full compromise.
* **Tool Redefinition ("Rug Pull"):** A tool that appears benign during initial validation could be maliciously altered after deployment to perform harmful actions.
* **Update Phase:**
* **Version Drift:** The continued use of older, vulnerable versions of the MCP protocol or server implementations leaves systems exposed to known exploits.
* **Privilege Persistence:** An attacker might retain elevated roles or old token scopes even after a system update, allowing for continued unauthorized access.
* **Unsigned Tool Manifests:** Tool manifests that are not cryptographically signed can be altered or injected post-deployment, allowing malicious tools to be introduced into the system.

This lifecycle-based threat model provides a clear map of where security controls are needed, informing the design of a defense-in-depth strategy.

3.2 COMPARATIVE FRAMEWORK FOR AGENT INTEROPERABILITY PROTOCOLS

MCP is one of several emerging protocols aimed at solving the agent interoperability challenge. A comparative survey by Ehtesham, A., et al. (2025) provides a valuable framework for understanding the landscape by analyzing four key protocols.1

3.2.1 AGENT COMMUNICATION PROTOCOL (ACP)

ACP is defined as a REST-native performative messaging layer designed for local multi-agent systems. It supports asynchronous streaming and multimodal agent responses through multi-part messages. Its primary focus is on bridging interoperability gaps between heterogeneous agents built on different stacks within an enterprise environment.1

3.2.2 AGENT-TO-AGENT PROTOCOL (A2A)

A2A is a peer-to-peer framework that facilitates task outsourcing and collaboration between distinct agentic systems. It uses capability-based "Agent Cards" over HTTP and Server-Sent Events (SSE) for enterprise-scale task orchestration. An agent discovers the capabilities of another by inspecting its Agent Card and then delegates tasks accordingly.1

3.2.3 AGENT NETWORK PROTOCOL (ANP)

ANP is a decentralized protocol designed for open-network agent discovery and secure collaboration. It is built on decentralized identifiers (DIDs) and JSON-LD graphs, making it suitable for creating open-internet agent marketplaces where agents from different organizations can discover and interact with each other securely.1

3.3 ANALYSIS OF SECURITY POSTURES AND MECHANISMS ACROSS PROTOCOLS

Each of these protocols embodies a different architectural and security philosophy, which is crucial for understanding why a one-size-fits-all security solution is inadequate. MCP's client-server architecture centralizes tool access through a server, creating a natural chokepoint for security monitoring and policy enforcement. In contrast, A2A's peer-to-peer model is more decentralized, meaning security must be enforced at the level of each individual agent. ANP takes decentralization a step further by relying on cryptographic DIDs for identity, building security on a foundation of verifiable credentials rather than a centralized authority.

This comparative analysis highlights the unique security posture of MCP. Its centralized server model is both a strength and a weakness: it provides a logical place to implement a security gateway, but it also represents a single point of failure if compromised. This justifies the need for a dedicated, robust security framework tailored to the specific threats of the MCP lifecycle, as the security mechanisms inherent in other protocols (like the decentralized identity of ANP) are not applicable. The analysis also reinforces the gap identified by Ehtesham et al. (2025) regarding the need for standardized benchmarks and "protocol bridges" to connect these disparate systems securely.1 The following table provides a succinct comparison of these protocols.

**Table 3.1: Comparative Analysis of Agent Interoperability Protocols**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dimension** | **Model Context Protocol (MCP)** | **Agent Communication Protocol (ACP)** | **Agent-to-Agent Protocol (A2A)** | **Agent Network Protocol (ANP)** |
| **Architecture** | Client-Server | REST-native, Broker-based | Peer-to-Peer | Decentralized Graph |
| **Communication** | JSON-RPC, HTTP, SSE | RESTful, Multi-part messages | HTTP, SSE, JSON-RPC | Encrypted, based on JSON-LD |
| **Discovery** | Server advertises capabilities | Centralized Agent Registry | Agent Cards at well-known endpoints | Decentralized discovery via DIDs |
| **Security Model** | Not specified; relies on transport layer (TLS) and implementation | Policy enforcement at ACP Server | Relies on standard web security (TLS, OAuth) | Based on Decentralized Identifiers (DIDs) |
| **Primary Use Case** | Standardized tool invocation and context ingestion for LLMs | Local multi-agent systems with multimodal messaging | Enterprise-scale collaborative task execution | Open-internet agent marketplaces and discovery |
| **Data Format** | JSON-RPC 2.0 | REST with JSON | JSON-RPC 2.0 | JSON-LD |

This comparison reveals that the four protocols are not necessarily direct competitors but may represent different layers or contexts of agent interaction. The phased adoption roadmap proposed by Ehtesham et al. (2025)—beginning with MCP for local tool access and extending to ANP for decentralized marketplaces—suggests a future where agents may need to utilize multiple protocols simultaneously.1 This multi-protocol reality implies that security solutions should be designed with modularity and adaptability in mind. While the framework proposed in this research is initially targeted at MCP, its core principles of dynamic trust and policy enforcement could be architected to be extensible to other protocols, making it a forward-looking and highly valuable contribution to the field of agent security.

**Chapter 4**

**DESIGN OF AN INTEGRATED, TRUST-AWARE SECURITY FRAMEWORK**

This chapter presents the core of the research: the detailed architectural design of the proposed integrated security framework. Translating the high-level concepts from the initial project review into a concrete blueprint, this design is grounded in established security principles and tailored to address the specific MCP threat model developed in the previous chapter. The framework's novelty lies in its unification of dynamic trust computation with task allocation, creating a system where security and operational efficiency are mutually reinforcing.

4.1 ARCHITECTURAL BLUEPRINT: A MODULAR, ZERO-TRUST APPROACH

The proposed security framework is architected with a modular design and is fundamentally based on the principles of Zero Trust Architecture (ZTA). The choice of ZTA, which operates on the maxim "never trust, always verify," is essential for the dynamic, open, and potentially untrusted nature of MCP environments.1 In an ecosystem where agents and tools can be added or modified at any time, the traditional notion of a trusted internal network becomes obsolete. ZTA mandates that every access attempt must be continuously validated, scrutinizing each interaction regardless of its origin.

4.1.1 THE MCP SECURITY GATEWAY: A CENTRALIZED POLICY ENFORCEMENT POINT

At the heart of the framework is the MCP Security Gateway, which serves as the central nervous system and a unified policy enforcement point. All tool execution requests from any agent are routed through this gateway. This centralized chokepoint is responsible for a sequence of critical security checks before granting execution: it verifies the agent's identity, validates the integrity of the requested tool's manifest, evaluates the agent's current trust score against the required threshold for the task, and enforces all relevant security policies.1 This design provides a single point of control and auditability for all agent actions within the system.

4.1.2 CORE COMPONENTS: IDENTITY, TRUST, AND POLICY MANAGEMENT

The framework is composed of three primary logical modules that work in concert within the Security Gateway:

1. **Identity & Trust Management:** This module is responsible for managing agent identities, potentially through a CA-style system, and, most critically, for dynamically calculating and updating the multi-dimensional trust score for each agent based on its real-time behavior.1
2. **Policy Engine:** This is a flexible, rule-based engine that governs all agent actions. It ingests the agent's identity, its trust score, and the risk score of the requested tool to make a GRANT or DENY decision based on a configurable set of policies.1
3. **Secure Tool Lifecycle Management:** This module oversees the entire lifecycle of MCP tools, from their initial registration and verification to their ongoing risk assessment and integrity monitoring, ensuring that only vetted and secure tools are available for execution.1

4.2 DYNAMIC TRUST COMPUTATION AND REPUTATION SYSTEMS

The computation of a reliable and expressive trust score in real-time is a primary research challenge addressed by this framework.1 The proposed system moves beyond simple binary trust models to a dynamic, multi-dimensional approach.

4.2.1 A MULTI-DIMENSIONAL TRUST MODEL

The trust score is calculated as a composite metric derived from several weighted factors, synthesizing concepts from established trust and reputation literature:

* **Behavioral History:** This factor is based on the agent's direct, observable performance. It includes metrics such as the success-to-failure ratio of past tasks, adherence to security policies (e.g., no failed authentication attempts), and resource consumption patterns.
* **Reputation (Indirect Trust):** Drawing from the global trust models surveyed by Yu, H., et al. (2013), this factor incorporates feedback and recommendations from other agents in the system.1 This allows the system to leverage collective intelligence to identify untrustworthy agents more quickly.
* **Competence (Functional Trust):** As distinguished by Yu, H., & Singh, H. K. (2002), this factor measures an agent's demonstrated capability for specific types of tasks.1 An agent may be highly trusted for data analysis tasks but have a low trust score for tasks requiring file system modification.
* **Task Criticality:** The trust required for a task is not static. Following the model proposed by Hang, C., & Wang, Y. (2007), the framework incorporates the criticality or sensitivity of the requested task into the final decision.1 A low-risk task may require a lower trust score than a high-risk one.

This multi-dimensional model provides a nuanced and context-aware measure of trustworthiness, allowing for more granular and intelligent access control decisions.

4.2.2 ADVANCED CONCEPTS: BLOCKCHAIN FOR IMMUTABLE REPUTATION

To further enhance the integrity of the trust system, this research considers the application of blockchain technology to create a tamper-proof and auditable ledger for agent reputation data. As proposed in works by Li, J., et al. (2021) and Wang, Y., et al. (2023), recording trust-related events (e.g., task successes, failures, policy violations) on a distributed ledger would make the reputation history immutable and resistant to manipulation by malicious agents.1 However, a critical analysis of this approach must also acknowledge its significant trade-offs. The performance overhead, in terms of both latency and computational cost, associated with blockchain transactions can be a major bottleneck for real-time MAS, where low-latency decision-making is paramount.1 Therefore, its application might be best suited for high-assurance systems where integrity guarantees outweigh performance considerations.

4.3 TRUST-AWARE TASK ALLOCATION AND ACCESS CONTROL ENGINE

The computed trust score is not merely a passive metric; it is the primary input that drives the operational logic of the task allocation and access control engine.

4.3.1 POLICY DEFINITION AND ENFORCEMENT

The Policy Engine operationalizes trust by enforcing a set of defined rules. These policies are expressed in a clear, human-readable format and directly link an agent's security posture to its operational capabilities. Examples of such policies include:

* ALLOW if agent.trust\_score > 0.8 AND tool.risk\_score < 0.5
* DENY if tool.permissions.contains("file\_system\_write") AND agent.trust\_score < 0.95
* REQUIRE\_HUMAN\_APPROVAL if task.criticality == "high" AND agent.competence[task.type] < 0.7

By making the trust score a central variable in these rules, the framework ensures that security is not an afterthought but is woven into the very fabric of agent coordination and task allocation.

4.3.2 MITIGATION STRATEGIES FOR SEMANTIC-LEVEL THREATS

The framework implements specific controls within the gateway to counter the modern, semantic-level threats identified in the literature:

* **Tool Poisoning Defense:** The gateway enforces strict validation of all tool manifests against a secure, cryptographically signed registry. It sanitizes tool descriptions to remove or encode potentially active content and employs semantic analysis and pattern matching (e.g., using YARA rules) to detect known malicious instruction patterns or command injection attempts within the descriptions.1
* **Backdoor Attack Defense:** To counter covert backdoors, the gateway monitors agent interaction patterns and tool invocation sequences for anomalous behavior. It also uses cryptographic attestation checks to ensure the integrity of a tool's binary or code before execution, preventing the use of tampered or redefined tools.1
* **Data Exfiltration Prevention:** The framework integrates principles from Data Loss Prevention (DLP) by inspecting the outputs of tool executions before they are returned to the agent. This allows the gateway to detect and block the leakage of sensitive data, such as personally identifiable information (PII) or credentials, based on predefined patterns.1

This design creates a virtuous cycle within the system. Agents that exhibit good behavior—successfully completing tasks without violating policies—will see their trust scores increase over time. A higher trust score, in turn, grants them access to more critical tasks and resources, making them more effective and integral to the system's operation. Conversely, malicious or poorly performing agents will have their trust scores penalized, which naturally isolates them and restricts their capabilities. This incentive structure transforms security from a purely restrictive function into a dynamic mechanism that promotes overall system efficiency, reliability, and cooperation, directly addressing the critical gap of separating security from task allocation.

**Chapter 5**

**IMPLEMENTATION AND FRAMEWORK INTEGRATION**

This chapter details the practical implementation of the proposed security framework, transitioning from architectural design to a functional prototype. It describes the technical realization of the core components and, critically, demonstrates the framework's real-world applicability through its integration with popular Multi-Agent System (MAS) orchestration frameworks. This focus on integration showcases the non-intrusive and adaptable nature of the design.

5.1 PROTOTYPING THE MCP SECURITY GATEWAY

The MCP Security Gateway, the central enforcement point of the framework, is implemented as a middleware service that intercepts all network traffic between MCP clients and servers. Functioning as a reverse proxy, it ensures that no agent can directly communicate with a tool-providing server without undergoing security validation.

The technical stack for the prototype consists of a lightweight web server that exposes an MCP-compliant endpoint. The core logic within the gateway is divided into its constituent modules:

* **Identity and Trust Management:** Agent identities are managed via a simple registry, with each agent assigned a unique identifier and cryptographic key pair for signing requests. Trust scores are stored in a key-value database (e.g., Redis) for fast retrieval and are updated asynchronously by a separate monitoring service that analyzes the audit log.
* **Policy Engine:** The policy engine is implemented using Open Policy Agent (OPA), a standard for unified policy enforcement. Security policies are written in OPA's declarative language, Rego, which allows for complex rules based on the agent's trust score, tool risk, and other contextual data. When a request is received, the gateway queries the OPA engine with the request metadata to receive a GRANT or DENY decision.

5.2 SECURE TOOL LIFECYCLE MANAGEMENT WORKFLOW

A robust security posture begins with ensuring that the tools available to agents are themselves secure. The framework implements a comprehensive workflow for managing the entire lifecycle of MCP tools, as outlined in the project scope 1:

1. **Registration:** Developers submit new tools to a secure registry via a dedicated portal. The submission includes the tool's manifest (defining its purpose, parameters, and required permissions) and its source code or binary.
2. **Verification:** Upon submission, an automated verification pipeline is triggered. This includes static analysis security testing (SAST) to scan the code for known vulnerabilities and dynamic analysis security testing (DAST) where the tool is executed in an isolated sandbox to observe its behavior and ensure it matches the description in its manifest.
3. **Risk Scoring:** A risk score is automatically assigned to the tool based on a predefined rubric. Factors influencing the score include the sensitivity of the permissions it requires (e.g., file system access is higher risk than a simple calculation), its external network dependencies, and the results of the security verification scans.
4. **Attestation:** Once a tool is approved, its manifest is cryptographically signed by a central authority. The MCP Security Gateway will only allow the execution of tools that present a valid, signed manifest, ensuring their integrity and preventing tampering or "rug pull" attacks where a tool's functionality is maliciously altered after approval.1

5.3 DEVELOPING ADAPTERS FOR MAS ORCHESTRATION FRAMEWORKS

A key objective of this research is to ensure the security framework can be seamlessly integrated into existing developer workflows without requiring a complete rewrite of agent logic. This is achieved by creating lightweight, "runnable adapters" for popular MAS orchestration frameworks.1 This adapter-based strategy decouples the security framework from the specific implementation details of any single agent framework, ensuring broad applicability and future-proofing the solution against a rapidly evolving ecosystem.

5.3.1 INTEGRATION WITH LANGGRAPH

LangGraph builds agentic workflows as cyclical graphs of nodes. To integrate the security framework, a custom wrapper function is created for tool-executing nodes. Instead of calling the tool directly, the node invokes the wrapper, which constructs a request to the MCP Security Gateway. The wrapper passes the agent's identity, the tool request, and any relevant context to the gateway. It then waits for the gateway's response, only proceeding with the actual tool execution if a GRANT decision is received. This allows security to be injected transparently at the graph level.

5.3.2 INTEGRATION WITH AUTOGEN

AutoGen from Microsoft facilitates conversations between multiple agents. Integration is achieved by creating a custom proxy agent or modifying the UserProxyAgent. When an agent in the AutoGen ecosystem needs to execute a tool or code, its request is not executed directly. Instead, it is passed to the custom proxy agent, whose sole responsibility is to communicate with the MCP Security Gateway. This proxy agent forwards the request for validation and only returns the execution result to the original agent after receiving approval from the gateway, effectively making the security framework a participant in the agent conversation.

5.3.3 INTEGRATION WITH CREWAI

CrewAI uses a high-level abstraction for defining agents, tasks, and crews. To integrate the security framework, the Tool objects used by CrewAI agents are customized. A new base class for tools, SecureTool, is created. When an agent invokes a method on a SecureTool object, the method's implementation first makes a call to the MCP Security Gateway for authorization. The actual tool logic is only executed if the gateway approves the request. This allows developers to continue defining tools in a familiar way while automatically inheriting the security controls.

5.4 SECURE EXECUTION, AUDITING, AND PERSISTENT MEMORY

The end-to-end runtime flow demonstrates the framework in action. A tool request initiated by an agent in a framework like LangGraph is intercepted by the corresponding adapter. The adapter forwards the request to the MCP Security Gateway, which authenticates the agent, retrieves its trust score, fetches the signed manifest and risk score for the requested tool, and evaluates the request against the policies in the OPA engine.

Upon a GRANT decision, the gateway executes the tool and returns the result. Critically, every step of this process—from the initial request to the policy decision and the final outcome—is logged to an immutable audit trail.1 This provides a detailed, tamper-proof record of all security-relevant events, which is essential for forensic analysis and for the continuous updating of agent trust scores.

Furthermore, the framework's principles can be extended to secure persistent memory. By integrating with a secure memory protocol like SAMEP 1, the gateway can enforce trust-based access controls not only on tool execution but also on the reading and writing of shared context. This ensures that an agent's access to sensitive information stored in long-term memory is also governed by its trustworthiness, creating a holistic security posture that covers both action and knowledge.

**Chapter n**

< Times New Roman, Font 12, Bold>

**Conclusion and Future Work**

< Times New Roman, Font 16, Bold>

This chapter should summarize the key aspects of your project (failures as well as successes) and should state the conclusions you have been able to draw. Outline what you would do if given more time (future work). Try to pinpoint any insights your project uncovered that might not have been obvious at the outset. Discuss the success of the approach you adopted and the academic objectives you achieved. Avoid meaningless conclusions, [e.g. NOT “I learnt a lot about C++ programming”]. Be realistic about potential future work. Avoid the dreaded: “All the objectives have been met and the project has been a complete success”. You have to crisply state the main take-away points from your work. Describe how your project is performed against planned outputs and performance targets. Identify the benefits from the project. Be careful to distinguish what you have done from what was there already. It is also a good idea to point out how much more is waiting to be done in relation to a specific problem, or give suggestions for improvement or extensions to what you have done.

Future scope of the work for improvement may also be included

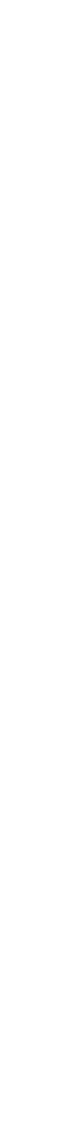
2

**Appendices**

< Times New Roman, Font 12, Bold>

Appendices are provided to give supplementary information, which is not included in the main text may serve as a separate part contributing to main theme.

* + - Appendices should be numbered using Arabic numerals, e.g. Appendix 1, Appendix 2 etc.
    - Appendices, tables and references appearing in appendices should be numbered and referred to at appropriate places just as in the case of chapters.
    - Appendices shall carry the title of the work reported in it and the same title shall be used in the contents page also.



**REFERENCES**

<Times New Roman, Font 16, Bold, CAPS>

// (ALL REFERNCE SHOULD BE IN APA FORMAT)

[1]. Eckart, C. (1951). Surface waves on water of variable depth, Wave Rep. 100, Scripps Inst. of Oceanogr., Univ. of California, pp.99. <For referring reports> <Times New Roman, Font 12>

[2]. Hasselmann, K., W.H. Munk and G.J.F. MacDonald (1963). Bispectra of Ocean Waves, In: M. Rosenblatt (ed.), Time Series Analysis, John Wiley Sons, p. 125, NewYork. <For referring from books edited from a collection of different papers>. <Times New Roman, Font 12>

[3]. Stoker, J.J. (1957). Water Waves. Interscience, New York. p. 520.

<For referring book> <Times New Roman, Font 12>

[4]. Tatavarti, Rao V.S.N. and D.A. Huntley (1987). Wave reflection at Beaches. Proc. Canadian Coastal Conference, Quebec City, pp. 241-255, Canada. <Times New Roman, Font 12>

<For referring conference proceedings>

[5]. Wallace, J.M. and R.E. Dickinson (1972). Empirical orthogonal representation of time series in the frequency domain, Part I: Theoretical considerations, J. App. Meteorology, Vol. 11, No. 6, pp. 887-892. <For referring journals> <Times New Roman, Font 12>