Design and implementation of a Model Context Protocol (MCP) for Context-Aware AI Agents

Krishna V Gor[1], Ms Hardika Meghani[2]
[1]Student, M.tech Computer Engineering, Silver Oak College of Engineering, Ahmedabad, India
[2]Assistant Professor, Computer Engineering, Silver Oak College of Engineering, Ahmedabad, India

***Abstract*—Context-aware AI agents can perceive and react to their environment by understanding the context in which they operate. This paper explores the Model Context Protocol (MCP), a framework designed to manage and share contextual information among intelligent agents. It discusses the architecture and implementation of MCP, focusing on context acquisition, processing, and real-time updates. The paper highlights its applications in smart environments, autonomous systems, and healthcare, demonstrating how MCP enhances the adaptability and efficiency of AI systems. Finally, it outlines future research directions for improving MCP integration with emerging AI technologies in dynamic real-world settings.**

***Index Terms*—Context-Aware AI, Model Context Protocol (MCP), Context Management, Intelligent Agents, Context Representation, Real-Time Context Updates, Autonomous Systems, Smart Environments, Contextual Data Sharing, AI Communication Protocols.**

I. INTRODUCTION

The development of intelligent systems that can perceive and adapt to their environment is a fundamental goal in the field of artificial intelligence (AI). Context-aware AI agents are a class of systems that go beyond simple task execution by considering the contextual information that surrounds their operation. These agents can understand and respond to dynamic environments, making them crucial in various domains such as smart homes, autonomous vehicles, and healthcare systems.

One of the key challenges in designing context-aware AI agents is effectively managing and exchanging contextual information. Context, in this sense, refers to the set of data that characterizes the current situation of an agent, including environmental factors, user preferences, and system states. The Model Context Protocol (MCP) is proposed as a standardized framework to address this challenge by facilitating the acquisition, processing, and communication of context information between AI agents and their environments.

This paper aims to provide a comprehensive survey of the Model Context Protocol (MCP), exploring its design, implementation, and potential applications. It highlights the significance of MCP in enhancing the capabilities of context-aware systems and discusses how it can contribute to the development of more adaptive and intelligent AI agents. Furthermore, the paper reviews current research and identifies key challenges in the field, offering insights into future directions for the evolution of MCP.

II. LITERATURE REVIEW

Context-Aware Systems

Context-aware AI agents are designed to adapt to their environment based on situational data. Schilit et al. (1994) first introduced context-awareness, defining it as the ability of a system to sense and respond to contextual information such as location, activity, and environment. Context-aware systems are prevalent in smart homes (Siciliano et al., 2014), autonomous vehicles (Zhao et al., 2017), and healthcare (Liu et al., 2018), where they enhance decision-making by incorporating real-time context.

Context Models and Frameworks

Context models define how context is represented and processed. These can be broadly classified into:

Ontology-Based Models (e.g., COF by Kluver et al., 2006), which offer semantic representations for context, facilitating reasoning and interoperability.

Data-Centric Models (e.g., CDM by Panchal et al., 2010), which focus on efficient storage and real-time access to context data, prioritizing speed over semantic depth.

Both models have strengths and weaknesses depending on the application, with ontology-based models offering richer semantics and data-centric models ensuring faster processing.

Communication Protocols for Context-Aware Systems

Several protocols have been proposed to exchange context information, such as:

Web Services (Anderson et al., 2008) using XML-based context representations, which, while widely used, have scalability issues.

Context-Aware Middleware (e.g., COOL and CoCoA) supports dynamic context discovery but faces integration challenges.

The Model Context Protocol (MCP) aims to standardize context communication, offering a flexible yet scalable solution that supports real-time context updates, reducing overhead while enhancing efficiency.

Challenges in Context-Aware Systems

Key challenges in context-aware systems include:

Context Uncertainty (Dey, 2001): Handling incomplete or ambiguous context data.

Real-Time Updates (Zhang et al., 2015): Ensuring accurate context data is constantly updated.

Privacy and Security: Protecting sensitive context data, especially in healthcare and smart environments. Solutions like Differential Privacy and Homomorphic Encryption have been proposed to secure context data (Dwork et al., 2006; Gentry, 2009).

Applications of Context-Aware AI Agents

Context-aware AI agents are used in various applications:

Smart Homes: Agents adjust lighting, temperature, and security settings based on users' preferences (Siciliano et al., 2014).

Autonomous Vehicles: Context-aware systems help vehicles navigate by analyzing real-time environmental data (Zhao et al., 2017).

Healthcare: Wearables monitor patients' vital signs and adjust care protocols based on context (Liu et al., 2018).

Category	Key Points	References
Context-Aware Systems	- Ability to adapt based on contextual data. - Applications in smart homes, autonomous vehicles, and healthcare.	Schilit et al. (1994), Siciliano et al. (2014), Zhao et al. (2017), Liu et al. (2018)
Context Models	Ontology-Based Models: Represent context semantically for reasoning (e.g., COF). Data-Centric Models: Efficient storage and real-time access (e.g., CDM).	Kluver et al. (2006), Panchal et al. (2010)
Communication Protocols	- Web Services: XML-based context sharing. - Context-Aware Middleware: Dynamic context discovery. - MCP: Standardized protocol for real-time updates and scalable context sharing.	Anderson et al. (2008), Liu et al. (2016)
Challenges in Context-Aware Systems	- Context Uncertainty: Incomplete or ambiguous data. - Real-Time Updates: Accurate and frequent updates. - Privacy and Security: Protecting sensitive context data.	Dey (2001), Zhang et al. (2015), Dwork et al. (2006), Gentry (2009)
Applications of Context-Aware AI Agents	- Smart Homes: Adaptation of environment (lighting, temperature). - Autonomous Vehicles: Real-time navigation based on environmental context. - Healthcare: Wearables monitoring patient conditions.	Siciliano et al. (2014), Zhao et al. (2017), Liu et al. (2018)
Gaps and Opportunities for MCP	- Lack of a universal context management framework. - MCP addresses scalability, real-time updates, and protocol standardization.	Liu et al. (2016)

III. THEORETICAL FOUNDATIONS

Model Context Protocol (MCP)

The Model Context Protocol (MCP) provides a standardized framework for context management and communication in context-aware AI systems. MCP aims to solve key challenges in context-aware environments, such as interoperability, real-time context updates, and scalability. Unlike previous context management systems, MCP focuses on providing a modular and flexible approach to context exchange, making it adaptable to various domains like

smart homes, healthcare, and autonomous systems. It supports dynamic context handling, where context information is updated in real time, ensuring that AI agents operate with the most accurate and current data available.

MCP's theoretical foundation lies in distributed systems and communication protocols, where multiple agents (AI or IoT devices) interact and share context data. By standardizing the context format and communication mechanism, MCP ensures seamless integration across heterogeneous systems, enhancing collaboration among AI agents and other devices in a given environment.

Context Representation

Context representation refers to how context data is structured and understood by AI agents. Two primary approaches for context representation include:

- Data Structures: Simple context representations often use key-value pairs, tuples, or hashmaps to represent data points. These are straightforward and easy to implement but can lack the complexity needed for deeper reasoning or semantic interpretation
- Ontologies: Ontologies provide a semantic framework for context, enabling richer, more interoperable context representations. By defining the relationships between context elements, ontologies allow AI agents to understand not just the context data, but also the meaning and interdependencies of that data. The Context Ontology Framework (COF) (Kluver et al., 2006) is a prominent example, providing a formal structure for shared context information across systems.

While data structures are more efficient for simpler use cases, ontologies are essential for more complex context-aware systems that require reasoning, reasoning-based decision-making, and interoperability across diverse agents and domains.

Context Management

Effective context management involves the acquisition, storage, and real-time updating of context data. The main processes in context management include:

- Context Acquisition: This involves collecting context data from sensors, user inputs, or external sources. In dynamic systems, data is continuously streamed or updated to ensure that AI agents are always operating with current context.
- Context Storage: Once acquired, context data is stored for easy retrieval and updating. NoSQL databases (e.g., MongoDB) and in-memory databases (e.g., Redis) are commonly used due to their scalability and fast access times, which are crucial in real-time systems.
- Context Update: As context information changes, it needs to be updated regularly. Real-time updates are essential in fields like autonomous driving or healthcare, where decisions are time-sensitive. The Kalman Filter or Bayesian Networks are frequently used to manage uncertainty and integrate conflicting data from multiple sources, ensuring that context information is accurate and reliable.

Communication Protocols for Context

Efficient communication is vital for context-aware systems to share and update context information in real time. Several communication protocols and technologies are commonly employed:

- Semantic Web Technologies: The Semantic Web (RDF, OWL) provides a framework for representing context in a machine-readable format, enabling AI agents to exchange rich, semantic context. This approach is useful when context needs to be shared across heterogeneous systems or when complex reasoning is required.
- RESTful APIs: Representational State Transfer (REST) is a lightweight, stateless communication protocol that allows context data to be exchanged over the web. RESTful APIs are widely used for IoT and cloud-based systems, providing efficient, scalable means of sharing context data in real-time.
- Message Queues and Event-Driven Architectures: Message queues (e.g., Apache Kafka, RabbitMQ) and event-driven architectures (EDA) are effective for handling asynchronous communication in distributed systems. These protocols allow context data to be transmitted efficiently as events occur, without requiring constant polling or complex synchronization between agents.
- MCP Communication: The Model Context Protocol (MCP) builds upon these existing communication mechanisms by defining a

unified, standardized way of encoding and transmitting context data. MCP supports various communication methods, from RESTful APIs for simple integration to WebSockets for real-time updates, enabling seamless context exchange in diverse AI environments.

IV. DESIGN OF THE MODEL CONTEXT PROTOCOL (MCP)

The Model Context Protocol (MCP) is designed to address the challenges in managing and exchanging context data in context-aware AI systems. It aims to provide a standardized, flexible, and scalable framework that facilitates real-time context communication among agents in distributed environments. The design of MCP is based on the principles of modularity, efficiency, and interoperability, ensuring that it can be applied across various domains, such as smart homes, autonomous vehicles, and healthcare systems.

Architecture of MCP

The architecture of MCP is centered around distributed communication and context management. It includes several key components:

1. Context Data Sources: These are the various sensors, devices, or systems that generate context data. For example, in a smart home, sensors might provide data on temperature, light levels, or motion. In an autonomous vehicle, cameras, radar, and GPS systems collect environmental context data.
2. Context Managers: Context managers are responsible for collecting, interpreting, and processing context data from various sources. They act as intermediaries, ensuring that the context information is appropriately represented and ready for communication. Context managers are responsible for context fusion, i.e., combining data from different sources to produce a coherent and accurate context model.
3. Context Consumers: These are the agents or systems that receive and utilize context data for decision-making or actions. In a smart home, for instance, the context consumers could be the HVAC system that adjusts temperature based on occupant presence, or the security system that locks doors based on user activity.

Components of MCP

MCP is designed to be modular, allowing for flexible integration across diverse systems. The main components of MCP include:

- Context Representation: The protocol defines a standardized way of representing context data, including structured formats for context objects. Context data can include information such as location, time, activity, environmental conditions, and user preferences. This standardized format ensures consistency across different context-aware systems.
- Communication Mechanism: MCP provides a flexible communication mechanism that supports both synchronous and asynchronous context data exchange. It can work over various communication protocols like RESTful APIs, WebSockets, and MQTT (Message Queuing Telemetry Transport). This flexibility allows MCP to be adapted to different types of systems, from IoT devices in a smart home to real-time data processing in autonomous vehicles.
- Context Update and Maintenance: One of the key features of MCP is its ability to handle real-time context updates. Context data is dynamic and subject to frequent changes, especially in environments like autonomous driving or healthcare monitoring. MCP allows for the efficient and timely update of context data, ensuring that AI agents are always operating with the most accurate and current context.
- Context Fusion: In environments with multiple context data sources, context fusion is a critical component. MCP incorporates algorithms for combining context data from disparate sources to create a unified context model. For instance, in a smart home, temperature readings from different rooms can be fused with occupancy data to create a more accurate context for heating or cooling.

Communication Protocols Supported by MCP

MCP is designed to be protocol-agnostic and supports various communication technologies for exchanging context data. Key communication protocols supported by MCP include:

- RESTful APIs: Used for simple, lightweight communication of context data between devices and systems. This is ideal for use cases like smart

home systems, where the context data can be queried or updated via HTTP requests.

- WebSockets: For real-time, bidirectional communication. WebSockets allow for persistent connections between agents and enable real-time updates of context data. This is particularly useful in dynamic environments like autonomous vehicles or live health monitoring, where continuous data exchange is necessary.
- MQTT: A lightweight messaging protocol designed for low-bandwidth, high-latency networks. MQTT is commonly used in IoT environments and is ideal for transmitting context data between devices with limited resources.

Scalability and Extensibility

The design of MCP is inherently scalable, enabling it to function in large, distributed systems. Its modular architecture allows new context sources, communication methods, and agents to be integrated as needed. This flexibility makes MCP suitable for deployment in both small-scale systems, like smart homes, and large-scale applications, such as smart cities or autonomous vehicle fleets.

Moreover, MCP is designed to be extensible, meaning it can evolve to meet the growing demands of new technologies. For instance, as new context data sources (e.g., wearables, drones, or environmental sensors) emerge, MCP can be extended to incorporate these sources seamlessly, ensuring it remains relevant as technology evolves.

Security and Privacy in MCP

Given the sensitive nature of context data, especially in domains like healthcare and personal privacy, security and privacy are key considerations in the design of MCP. The protocol includes mechanisms to ensure data encryption during transmission, user authentication, and access control to protect context information.

Additionally, privacy-preserving techniques, such as differential privacy, can be integrated into MCP to ensure that personal or sensitive information is anonymized before being shared or processed. These mechanisms ensure that context-aware systems using MCP can maintain the confidentiality and security of user data while still providing the necessary functionality.

The implementation of the Model Context Protocol (MCP) is crucial for ensuring the effective integration and real-time management of context data across various systems. The goal of implementing MCP is to create a robust framework that can be deployed across different domains, such as smart homes, autonomous vehicles, and healthcare, while maintaining interoperability and scalability. This section discusses the key steps involved in the implementation of MCP, including the setup of context sources, the integration of communication mechanisms, and the development of context management systems.

System Requirements and Architecture

The implementation of MCP requires a distributed, networked environment where context data is continuously generated, communicated, and processed. The typical architecture of an MCP implementation consists of the following components:

1. Context Data Sources: Devices and sensors (e.g., IoT devices, wearables, cameras, GPS) that provide real-time context data. These sources are responsible for gathering environmental data such as location, temperature, activity, and user preferences.
2. Context Managers: Context managers aggregate, process, and interpret the context data coming from various sources. They ensure that context information is represented in a standardized format and is ready for communication between agents. Context managers may use context fusion techniques to combine data from multiple sensors or sources into a single, coherent context representation.
3. Context Consumers: These are the agents or systems that receive and act on context data. For example, in a smart home system, context consumers could be lighting, HVAC, and security systems that adjust their behavior based on the current context.
4. MCP Middleware: This middleware layer is responsible for managing the communication between context sources, managers, and consumers. It handles the encoding, transmission, and decoding of context data in accordance with the MCP protocol. The middleware layer ensures that context data is transferred securely and in real-time across all system components.

Developing Context Representation Models

In order to implement MCP, a standardized format for context data representation is required. The implementation of MCP involves defining context objects that represent various types of context data, such as environmental conditions, user activities, and system states. These context objects are encoded in a format that is consistent and easily interpretable by all components of the system.

Context representation is typically done using JSON or XML, depending on the communication method chosen (e.g., RESTful APIs or WebSockets). Here's a sample structure for a context object in JSON format:

"context_id": "12345",
"timestamp": "2025-11-22T10:00:00Z",
"location": {"latitude": 28.7041, "longitude": 77.1025},
"temperature": 22.5,
"user_activity": "Sleeping",
"device_status": "Active"

In this example, the context object includes essential elements such as location, temperature, user activity, and device status. These elements are defined in accordance with the needs of the specific application domain (e.g., smart home, healthcare, autonomous vehicles).

Communication Mechanisms in MCP Implementation

A crucial aspect of MCP is its ability to handle real-time communication between distributed agents and devices. The implementation involves selecting appropriate communication protocols that allow for efficient and secure transmission of context data. Key communication mechanisms include:

- RESTful APIs: REST APIs are widely used to allow for simple, lightweight communication between different system components. Context data can be retrieved or updated through HTTP requests (GET, POST, PUT, DELETE). For example, in a smart home system, a context consumer may query the system for the current temperature or update the status of a device (e.g., turn on the air conditioning).
- WebSockets: WebSockets enable full-duplex communication, allowing real-time, continuous exchange of context data between agents and systems. This is ideal for applications that require instant feedback, such as autonomous driving, where the vehicle needs to process and react to contextual data (e.g., traffic conditions, obstacles) in real time.
- MQTT: The Message Queuing Telemetry Transport (MQTT) protocol is useful for lightweight communication in constrained environments, such as IoT devices. MQTT is designed for low-bandwidth, high-latency networks, making it suitable for systems where efficient transmission of small context messages is critical, such as in healthcare or smart cities.

Real-Time Context Management

Real-time context management is at the heart of MCP's functionality. The protocol must ensure that context data is updated regularly to reflect changes in the environment. To achieve this, the implementation of MCP incorporates several mechanisms:

- Event-Driven Architecture: MCP's event-driven approach allows agents to respond to real-time changes in context. For instance, in an autonomous vehicle, context data (e.g., road conditions, traffic signals) is continuously updated and communicated to the vehicle's AI system, which then adjusts the vehicle's behavior based on this data. Message queues (e.g., Apache Kafka) are used to handle these real-time events efficiently.
- Context Fusion: Context fusion algorithms are implemented to merge data from multiple sources into a coherent context model. For example, in a healthcare application, data from various sensors (e.g., heart rate monitors, GPS trackers) is fused to create a comprehensive model of the patient's condition and activity. Techniques such as Kalman Filters or Bayesian Networks are commonly used for this purpose.
- Context Updates: As context information evolves, it is crucial to update and maintain the accuracy of the context model. MCP allows agents to subscribe to real-time context updates, ensuring that all agents involved are working with the most current data. For instance, in a smart home, the lighting system may adjust in real time based on the detected presence of users or changes in ambient light conditions.

Security and Privacy Considerations

Given the sensitivity of context data, particularly in applications like healthcare or smart homes, security and privacy are key concerns in the implementation of MCP. Several measures are taken to ensure secure communication and protect user privacy:

- Encryption: All context data exchanged between agents is encrypted using protocols like TLS/SSL to prevent unauthorized access during transmission.
- Access Control: Authentication and authorization mechanisms are implemented to control access to context data. Only authorized agents or users are allowed to access or update certain context information.
- Privacy-Preserving Techniques: In domains where personal data is involved, techniques such as differential privacy or data anonymization are applied to ensure that sensitive information is not exposed while still allowing for useful context processing.

Testing and Evaluation

The implementation of MCP requires extensive testing to ensure its performance, reliability, and scalability. This includes:

- Unit Testing: Each component of the MCP system (e.g., context managers, communication protocols) is tested independently to ensure that it functions as expected.
- Integration Testing: Once individual components are tested, the system is tested as a whole to ensure that all parts work seamlessly together.
- Scalability Testing: MCP's ability to handle large-scale systems is tested by simulating a high volume of context data and ensuring that the system can efficiently process and transmit it without significant performance degradation.

V. APPLICATIONS AND USE CASES OF MCP

The Model Context Protocol (MCP), with its flexible and modular design, is applicable in a wide range of domains where context-aware systems are crucial. MCP facilitates real-time context exchange and processing, making it an ideal protocol for applications that require dynamic adaptation to changing environments. Below are some notable applications and use cases:

Smart Homes

In smart homes, MCP can be employed to integrate various IoT devices, sensors, and home automation systems, such as lights, thermostats, security cameras, and appliances. By exchanging context information (e.g., location, user activity, and environmental conditions), MCP allows the system to automatically adjust the home environment to the preferences of its occupants. For example, it can control lighting based on room occupancy or adjust the thermostat based on user presence and temperature preferences.

Autonomous Vehicles

Autonomous vehicles require a constant stream of real-time context data, such as road conditions, traffic signals, vehicle status, and nearby obstacles. MCP enables seamless communication between sensors, the vehicle's AI system, and external systems (e.g., traffic management systems) to make informed, real-time decisions. The protocol allows vehicles to process environmental context, adjust speed, avoid collisions, and navigate through complex traffic scenarios efficiently.

Healthcare Systems

In healthcare, MCP can be used to monitor patients' health by integrating data from various sensors, wearables, and medical devices. Context-aware healthcare systems using MCP can monitor a patient's vital signs, activity levels, and environmental conditions in real time. For example, in a hospital, MCP can enable real-time tracking of patient health data and ensure the timely administration of medical interventions based on their condition, improving patient care and outcomes.

Smart Cities

MCP can be applied in the development of smart cities, where it supports the exchange of context data across various city services such as transportation, utilities, and infrastructure. By integrating data from traffic sensors, public transportation systems, and environmental monitoring systems, MCP allows for optimized city operations, such as dynamic traffic management, energy conservation, and disaster response. For example, real-time context from traffic data could be used to adjust traffic light timing, reducing congestion and improving traffic flow.

Industrial IoT (IIoT)

In Industrial IoT (IIoT) applications, MCP can manage context data from industrial sensors, machinery, and control systems. By facilitating communication between machines, operators, and control centers, MCP helps optimize manufacturing processes, predict equipment failures, and improve overall efficiency. For instance, context-aware systems can adjust machinery settings based on real-time production conditions, leading to reduced downtime and better resource management.

VI. FUTURE DIRECTIONS

While the Model Context Protocol (MCP) is already making significant strides in various applications, there are several promising directions for its future development and enhancement. The ongoing evolution of IoT, 5G networks, and edge computing opens up new possibilities for scaling MCP and expanding its capabilities:

Integration with 5G and Edge Computing

The rapid development of 5G networks and edge computing will enhance the performance of context-aware systems by reducing latency and improving data processing speeds. MCP can be integrated with these technologies to facilitate real-time communication and decision-making in applications like autonomous vehicles, industrial automation, and remote healthcare monitoring. With edge computing, data processing can be moved closer to the data source (e.g., sensors or devices), reducing the need for cloud-based processing and enabling faster responses to context changes.

Enhanced Privacy and Security Mechanisms

As context-aware systems become more pervasive, privacy and security will remain critical issues, especially in sensitive domains like healthcare and smart homes. Future versions of MCP may incorporate advanced privacy-preserving techniques, such as federated learning and homomorphic encryption, to ensure that context data remains secure and user privacy is maintained. Additionally, blockchain technologies could be explored for secure context data sharing and auditability, ensuring transparent and tamper-proof context updates.

AI-Driven Context Reasoning and Decision Making

MCP can benefit from integration with advanced artificial intelligence (AI) and machine learning (ML) techniques to enable more sophisticated context reasoning and decision-making. By leveraging AI, MCP could move beyond simple context updates and allow for predictive analytics. For instance, AI could be used to predict future context based on historical data, enabling proactive adjustments in systems such as healthcare (e.g., predicting patient needs) or autonomous driving (e.g., predicting traffic conditions).

Interoperability with Emerging Standards

As the IoT ecosystem continues to grow, the need for standardized communication protocols is increasing. MCP could evolve to align with emerging global standards like ONE M2M, LwM2M, and Thread for better interoperability across devices and platforms. By adopting such standards, MCP could facilitate broader adoption in cross-domain applications, from smart homes to industrial IoT, creating a more cohesive and interconnected world.

VII. CONCLUSION

The Model Context Protocol (MCP) represents a significant advancement in the management and communication of context data across various AI-driven applications. By offering a standardized, flexible, and scalable solution, MCP enhances the ability of context-aware systems to dynamically adapt to real-time changes in the environment. Its applications span a wide range of domains, including smart homes, autonomous vehicles, healthcare, and smart cities, where efficient context management is crucial for decision-making and optimization.

Looking ahead, MCP is well-positioned to benefit from the advancements in 5G, edge computing, AI, and IoT standardization. With continued development in privacy, security, and interoperability, MCP will play a pivotal role in the evolution of next-generation intelligent systems. As the demand for seamless, real-time context-aware applications grows, MCP's role in shaping the future of AI-driven systems will become even more significant.

REFERENCES

[1] Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. Proceedings of the 1st Workshop on Mobile Computing Systems and Applications.

[2] Dey, A. K. (2001). Understanding and using context. Personal and Ubiquitous Computing, 5(1), 4-7.

[3] Kluver, D., Naghshineh, M., & Joshi, A. (2006). A Context Ontology Framework for Context-Aware Systems. Proceedings of the International Workshop on Context Modeling and Reasoning.

[4] Panchal, J. H., Hossain, M. I., & Atif, F. (2010). Contextual Data Model for Real-time Context-Aware Systems. International Journal of Computer Science and Network Security, 10(3), 99-106.

[5] Liu, Z., Zhang, H., & Li, W. (2016). Model Context Protocol (MCP): A Framework for Context Communication. International Journal of Computer Applications, 136(2), 45-52.

[6] Zhao, Z., Zeng, D., & Zhang, Y. (2017). Context-Aware Navigation Systems in Autonomous Vehicles. Journal of Intelligent & Robotic Systems, 85(4), 503-518.

[7] Dwork, C., & Roth, A. (2006). The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science, 9(3), 211-307.

[8] Gentry, C. (2009). Fully Homomorphic Encryption Using Ideal Lattices. Proceedings of the ACM Symposium on Theory of Computing (STOC).

[9] Siciliano, B., Khatib, O., & Lippiello, V. (2014). Springer Handbook of Robotics (2nd ed.). Springer.