Universidad Miguel Hernández de Elche

MASTER UNIVERSITARIO EN ROBÓTICA

Enhancing Communication Security in ROS 2

Trabajo de Fin de Máster

2024/2025

Autor: Francisco Javier Blanco Romero

Tutor: José María Azorín Poveda

Summary

This master thesis explores the intersection of robotics, cybersecurity, and post-quantum cryptography, focusing on enhancing the communication security of the Robot Operating System 2 (ROS 2). As robotic systems become increasingly interconnected and deployed in sensitive applications, the need for robust security measures has never been more pressing. This research addresses the growing concern that current cryptographic methods, including those implemented in ROS 2's security framework, may be vulnerable to future quantum computing attacks.

The thesis begins with an overview of networking in robotics, emphasizing the unique challenges faced in teleoperation, telemetry, Robot-to-Robot communication (R2R) and networked robotics. It then analyzes ROS 2's communication architecture, examining the transition from ROS 1 and the adoption of the Data Distribution Service (DDS) as the primary middleware. The new ROS 2 alternative middleware, Zenoh, is also considered. Special attention is given to the security features of DDS and the Secure Robot Operating System 2 (SROS2) framework.

A significant portion of the research is dedicated to post-quantum cryptography (PQC) and its potential integration into ROS 2. The thesis presents a novel approach to enhancing ROS 2 security by designing and implementing a PQC plugin for DDS. This includes an exploration of adapting PQC for alternative middlewares, particularly Zenoh, which has emerged as a promising option for addressing some of the limitations of DDS in certain robotic applications. The practical implications of integrating PQC into ROS 2 are considered.

Finally, the thesis discusses the challenges encountered during the research, proposes potential improvements, and outlines future research directions in the field of secure robotic communications. The findings of this study contribute to the ongoing efforts to secure robotic systems against evolving cyber threats.

By addressing the need for quantum-resistant security in robotic communications, this research aims to enhance the security of ROS 2-based systems in various applications, from industrial automation to healthcare robotics.

Resumen

Este Trabajo de Fin de Máster explora la intersección de la robótica, la ciberseguridad y la criptografía post-cuántica, centrándose en mejorar la seguridad de las comunicaciones del Sistema Operativo de Robots 2 (ROS 2). A medida que los sistemas robóticos se interconectan cada vez más y se implementan en aplicaciones sensibles, la necesidad de medidas de seguridad sólidas es más importante que nuca. Esta investigación aborda la creciente preocupación de que los métodos criptográficos actuales, incluidos los implementados en el marco de seguridad de ROS 2, puedan ser vulnerables a futuros ataques de computación cuántica.

La tesis comienza con una descripción general de las redes en robótica, enfatizando los desafíos únicos que enfrentan la teleoperación, la telemetría, la comunicación robot-robot y las redes robóticas. Luego profundiza en un análisis de la arquitectura de comunicaciones de ROS 2, examinando la transición de ROS 1 y la adopción del Servicio de Distribución de Datos (DDS) como el middleware principal. También se considera Zenoh, un nuevo middleware alternativo para ROS 2. Se presta especial atención a las características de seguridad de DDS y el marco del Sistema Operativo de Robots Seguros 2 (SROS2).

Una parte importante de la investigación está dedicada a la criptografía post-cuántica (PQC) y su posible integración en ROS 2. La tesis presenta un enfoque novedoso para mejorar la seguridad de ROS 2 mediante el diseño e implementación de un complemento PQC para DDS. Esto incluye una exploración de la adaptación de PQC para middlewares alternativos, en particular Zenoh, que ha surgido como una opción prometedora para abordar algunas de las limitaciones de DDS en ciertas aplicaciones robóticas. Se consideran las implicaciones prácticas de la integración de PQC en ROS 2.

Finalmente, la tesis analiza los desafíos encontrados durante la investigación, propone posibles mejoras y describe futuras direcciones de investigación en el campo de las comunicaciones robóticas seguras. Los hallazgos de este estudio contribuyen a los esfuerzos en curso para proteger los sistemas robóticos contra las amenazas de ciberseguridad en evolución.

Al abordar la necesidad de seguridad resistente a la computación cuántica en las comunicaciones robóticas, esta investigación tiene como objetivo mejorar la seguridad de los sistemas basados en ROS 2 en varias aplicaciones críticas, desde la automatización industrial hasta la robótica sanitaria.

Contents

List of Figures

Glossary

DDS	Data Distribution Service - A middleware protocol for data-centric publish-subscribe communication in distributed systems.
DTLS	Datagram Transport Layer Security - A communications protocol that provides security for datagram-based applications.
ICE	Interactive Connectivity Establishment - A technique used in computer networking to find ways for two computers to talk to each other as directly as possible in peer-to-peer networking.
IoT	Internet of Things - The network of physical objects embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.
MQTT	Message Queuing Telemetry Transport - A lightweight publish-subscribe network protocol that transports messages between devices.
PQC	Post-Quantum Cryptography - Cryptographic algorithms that are thought to be secure against an attack by a quantum computer.
ROS	Robot Operating System - A flexible framework for writing robot software.
RTPS	Real-Time Publish-Subscribe - The wire protocol used by DDS to enable interoperability between different vendor implementations.
SROS2	Secure Robot Operating System 2 - A set of security tools and libraries for ROS 2.
TLS	Transport Layer Security - A cryptographic protocol designed to provide communications security over a computer network.
UDP	User Datagram Protocol - A communications protocol that is primarily used for establishing low-latency and loss-tolerating connections between applications on the internet.
VPN	Virtual Private Network - A service that allows you to create a secure connection to another network over the Internet.
Zenoh	A scalable and performant protocol for IoT and edge computing, used as an alternative middleware for ROS 2.

1 Introduction

1.1 Background and Motivation

The Robot Operating System (ROS) [1, 2] has emerged as a dominant framework in the robotics community, with ROS 2 positioning itself as a more robust and industry-oriented successor [3]. While this Master thesis focuses on ROS 2, it also considers broader applications in networked cyber-physical systems. These systems, which include but extend beyond traditional robotics, can be categorized into several key domains:

1. **Control Systems:** Encompassing both local and remote control of robotic and automated systems, including industrial automation and process control.

2. **Telerobotics:** Focusing on the remote operation of robots, particularly in scenarios requiring human intervention in distant or hazardous environments.

3. **Telemetry:** Addressing the collection and transmission of data from remote systems, essential for monitoring and analysis in various fields including aerospace, healthcare, and environmental monitoring.

4. **Multi-Robot Systems:** Exploring the communication and coordination aspects of multiple robotic units, including:

    - Robot-to-Robot (R2R) Communication: Direct interaction between individual robotic units.

    - Swarm Robotics: Collective behavior and communication in large groups of relatively simple robots.

The adoption of ROS 2 in sectors like healthcare, industrial automation, and autonomous vehicles has highlighted the need for robust security measures. The transition to ROS 2, particularly its use of the Data Distribution Service (DDS) middleware, while improving reliability and performance, has introduced new security challenges.

Cyber-physical systems, including robotic platforms, have become prime targets for cyber threats due to their direct influence on the physical world. The increasing interconnectivity of these systems, often extending to external networks and potentially the global Internet, significantly expands the attack surface. This trend towards greater networking, while beneficial for functionality, exposes these systems to a broader range of potential network-based attacks.

As the cyber threat landscape evolves, cybersecurity is becoming an important factor in technology selection for cyber-physical systems. Organizations are increasingly prioritizing solutions with strong security guarantees, especially in applications where breaches could have severe real-world consequences. This shift underscores the urgent need for security measures in ROS 2 and similar systems.

The importance and timeliness of robotics cybersecurity research are underscored by recent developments in both the industry and academia. The European Innovation Council (EIC) has recently awarded significant funding to Alias Robotics, a Spanish robotics cybersecurity company. This funding, comprising €2.5M in grants and €5M in investment, highlights the EU's commitment to advancing cybersecurity in robotics and related fields [4].

One of the most pressing concerns in the field of cybersecurity, including robotics security, is the threat of quantum computing. Since the seminal work by Shor and Grover [5, 6], advancements in quantum computing are estimated to pose a threat to current cryptographic methods within the next 10 to 20 years [7]. This timeline suggests that without adaptation, traditional asymmetric encryption protocols will become ineffective against quantum computing capabilities. To overcome this threat, the usage of post-quantum cryptography (PQC) or quantum-resistant algorithms is proposed and is being implemented [8].

The urgency of addressing quantum computing threats is further emphasized by leading cybersecurity agencies worldwide. Organizations such as the American NCSC, NSA, CISA, and NIST [9, 10, 11], along with European bodies like ENISA and national agencies including CCN (Spain) [12], have collectively stressed the need to transition towards PQC. Notably, a position paper [13] jointly issued by several European agencies strongly advocates for quantum-resistant cryptography as a more practical and cost-effective solution compared to alternatives like Quantum Key Distribution (QKD). This paper was authored by four key European cybersecurity agencies: the French Cybersecurity Agency (ANSSI), the German Federal Office for Information Security (BSI), the Netherlands National Communications Security Agency (NLNCSA), and the Swedish National Communications Security Authority (Swedish Armed Forces). Their stance has gained further support from the National Cyber and Information Security Agency (NÚKIB) of the Czech Republic [14]. These agencies emphasize the need for further research and development in PQC, highlighting its potential to address current shortcomings in quantum-safe communication strategies. Recently, another significant milestone in this transition was reached when NIST finalized standards for post-quantum cryptographic algorithms [15].

This need is also highlighted by recent EU proposals. The European Union has demonstrated a strong interest in advancing post-quantum cryptography and its applications in critical sectors. This is evidenced by two significant calls for proposals:

1. The Digital Europe Programme call "Deployment of Post Quantum Cryptography in systems in industrial sectors" (DIGITAL-ECCC-2024-DEPLOY-CYBER-06-PQCINDUSTRY), which focuses on enabling PQC adoption in industrial sectors such as automotive, automation, finance, control systems, and energy. This call emphasized the integration of standardized PQC protocols into existing digital security and communication networks [16].

2. The Horizon Europe call "Post-quantum cryptography transition" (HORIZON-CL3-2024-CS-01-02), which aims to increase the maturity of post-quantum cryptographic algorithms and contribute to their standardization. This call also seeks to develop tools for large-scale implementation of PQC algorithms and secure transition strategies from pre- to post-quantum encryption [17].

The confluence of these factors – the widespread adoption of ROS, the vulnerability of command and telemetry systems, the growing threat landscape for cyber-physical systems, the increasing interconnectedness of robotic networks, the rising importance of cybersecurity in technology decisions, and the significant EU investment in related fields – creates a compelling motivation for this research. By focusing on enhancing the communication security in ROS 2, this thesis aims to address a need in the field, contributing to the safety and reliability of a wide range of cyber-physical systems that rely on robust, secure communication frameworks.

1.2 Problem Statement

Despite the advancements in ROS 2, its current security framework, primarily based on DDS Security, may not be sufficient to counter emerging threats, particularly those posed by quantum computing. This vulnerability extends to ROS 2 systems, potentially exposing robotic applications to security risks. Therefore, there is a need to enhance the communication security in ROS 2 to ensure its resilience against both current and future threats.

1.3 Research Objectives

This thesis aims to address the security challenges in ROS 2 by integrating post-quantum cryptography (PQC) into its communication stack. The primary objectives of this research are:

1. To analyze the current security architecture of ROS 2, with a focus on its DDS-based communication layer and the emerging Zenoh middleware.

2. To investigate the feasibility and methods of incorporating post-quantum cryptographic algorithms into ROS 2's security framework, including SROS2.

3. To design and implement PQC enhancements for both DDS and Zenoh-based communications in ROS 2, exploring different approaches such as end-to-end encryption and transport-level security.

4. To explore and compare secure communication strategies when using Zenoh as a bridge for ROS 2 over wide-area networks and as a native middleware for ROS 2.

5. To identify current limitations and challenges in implementing PQC in ROS 2, particularly focusing on the constraints in Zenoh's cryptographic libraries and potential future developments.

1.4 Master Thesis Structure

This master thesis is organized as follows: Chapter 1 provides an introduction to the research, including background, motivation, problem statement, and research objectives. Chapter 2 offers an overview of networked robotics, discussing various applications and communication requirements in robotic systems. Chapter 3 explores the ROS communication stack, exploring middleware options, protocols, and internet connectivity solutions for ROS 1 and ROS 2. Chapter 4 focuses on security in ROS 2, covering general security considerations in robotics, post-quantum cryptography, ROS 2/DDS security, Zenoh security, and an analysis of SROS2. Chapter 5 presents the core contribution of this research, detailing the design and implementation of post-quantum cryptography integration in ROS 2, including extending SROS2 for PQC support, secure communication with the Zenoh bridge and middleware, and a discussion of current limitations and potential improvements. Finally, Chapter 6 concludes the thesis, summarizing the contributions, discussing implications for ROS 2 and robotics security, and outlining challenges and future work directions.

2 Networked Robotics: Applications and Communication Requirements

The field of robotics heavily relies on effective networking technologies to enable communication between various components of robotic systems, as well as between robots and their control stations. As robotic systems become increasingly complex and distributed, the importance of robust, efficient, and secure networking solutions has grown significantly.

In recent years, cyber-physical systems—computing systems that interact with and influence the physical world, including industrial control systems, medical devices, and robotic platforms—have become increasingly attractive targets for malicious actors. This trend underscores the importance of robust cybersecurity measures in robotic networking. As these systems become more interconnected and autonomous, the security of their communication channels becomes more important [18].

As we focus on communication security in the context of ROS 2, it is essential to review the applications and contexts in which communications appear in robotics. Key areas include:

- Control
- Teleoperation
- Telemetry
- Robot-to-Robot (R2R) Communication

2.1 Control Systems

Control systems in robotics involve the management and regulation of robotic behavior through communication networks [19, 20]. These systems are essential for ensuring precise and responsive robot operations, often requiring real-time data exchange between sensors, actuators, and control units.

In networked control systems, the communication infrastructure plays a major role in transmitting control signals and feedback data. The effectiveness of these systems depends heavily on the reliability, latency, and bandwidth of the underlying network.

Key aspects of control systems in robotics include:

- **SCADA in Robotics:** Supervisory Control and Data Acquisition (SCADA) systems for large-scale robotic deployments.
- **Real-time Control Networks:** Protocols and technologies for low-latency control communications.
- **Distributed Control Architectures:** Networked approaches to decentralized robot control.
- **Industrial Control Protocols:** Application of protocols like Modbus, Profinet, and EtherCAT in robotic systems.

- **Edge Computing in Control:** Leveraging edge devices for local processing and control.
- **Network-induced Challenges:** Addressing issues like jitter, packet loss, and delays in control systems.
- **Cyber-Physical Systems:** Integration of physical processes with networked computing in robotic control.
- **Adaptive Control over Networks:** Techniques for maintaining control stability in varying network conditions.

2.2 Telerobotics

Teleoperation [21, 22, 23], the practice of controlling robots from a distance, presents unique challenges and requirements for robotic networking. This field has gained significant traction in various domains, including space exploration, underwater research, military applications, and robotic surgery, where it has led to unprecedented advancements.

Teleoperated robotic systems typically comprise several main elements [23]:

- **Operator:** The person who performs remote control of the operation. Their action can range from continuous control to supervised control.
- **Slave:** The device located in the remote area that is being controlled by the operator. It can be a manipulator or a robot.
- **Interface:** Equipment and programs that communicate between the operator and the elements of the remote environment, such as the master manipulator, video monitors, voice recognition systems, etc.
- **Control and Communication Channels:** Devices responsible for transmitting and processing signals sent between the remote and local areas.
- **Sensors:** Devices that collect information from the local and remote areas to be used by the interface and control systems.

The operator interface serves as the point of interaction for human controllers, allowing them to issue commands and receive feedback. This interface can range from simple keyboard controls to sophisticated haptic feedback systems. The communication link, essential for transmitting data between the operator and the robot, must support bidirectional, often real-time data flow. The robot, whether fixed or mobile, houses the necessary mechanical and electronic components to execute commands and gather environmental data.

Key considerations in teleoperation include:

- **Bilateral Communication:** Teleoperation requires robust two-way data flow, not only for control commands but also for sensory feedback and system acknowledgments [18].
- **Real-time Requirements:** System stability often depends on strict real-time performance, with specific thresholds for maximum delay and packet loss [18]. For instance, in telesurgical applications, studies suggest that surgeons can adapt to certain latencies but rec-

ommend a maximum latency of approximately 300 ms [24].

- **Data Characteristics:** Control data in teleoperation is typically compact, ranging from 20 to 100 octets, but requires frequent transmission [18]. This high-frequency, low-volume data pattern presents unique networking challenges.

- **Quality of Service (QoS):** Teleoperation systems demand specific QoS parameters, including minimal latency, low jitter, and high reliability to ensure smooth and responsive control [25].

Wirz et al. [26] propose additional requirements for teleoperation protocols, including mechanisms for smooth congestion avoidance, services that differentiate and prioritize various data streams, and the provision of Round-Trip Time (RTT) feedback to the application layer for dynamic control parameter adjustments.

The design of teleoperated robots varies based on their specific applications, typically including components such as power systems, communication transceivers, embedded computation units, and various sensors. As teleoperation technology advances and expands into new domains, it continues to push the boundaries of remote robotic control. However, this expansion also brings new challenges, particularly in ensuring secure and reliable communication over varied and often unpredictable network conditions. Addressing these challenges remains a key focus in the ongoing development of teleoperation systems.

2.3 Telemetry

Telemetry in robotics involves the remote collection and transmission of data from robotic systems to monitoring stations or control centers. This process is essential for maintaining situational awareness, performing diagnostics, and enabling remote decision-making in robotic operations.

Vasseur and Dunkels [27] define telemetry as the process of performing remote measurements, derived from the Greek words *tele* (remote) and *metron* (to measure). They note that Machine-to-Machine (M2M) communication, a broader concept encompassing telemetry, can be applied in both long-distance and short-distance scenarios, from remote weather stations to medical devices.

Key aspects of telemetry in robotics include:

- **Data Types in Robotic Telemetry:** Sensor readings, status information, environmental data.

- **Telemetry Protocols:** Lightweight protocols for efficient data transmission (e.g., MQTT, CoAP).

- **Bandwidth Management:** Techniques for optimizing data transfer in limited-bandwidth environments.

- **Real-time vs. Batch Telemetry:** Trade-offs and use cases for different data transmission strategies.

- **Edge Computing in Telemetry:** Local processing to reduce data transmission needs.

- **Telemetry Security:** Ensuring data integrity and confidentiality during transmission.
- **Scalability in Robotic Swarms:** Managing telemetry data from large numbers of robots.
- **Integration with Control Systems:** Using telemetry data for real-time decision-making and control.
- **Long-range Telemetry:** Challenges and solutions for remote robotics operations.

Effective telemetry systems in robotics must balance the need for comprehensive data collection with the constraints of network capacity, power consumption, and real-time requirements. As robotic systems become more autonomous and operate in more diverse environments, the role of telemetry in maintaining operational awareness and enabling remote intervention becomes increasingly relevant.

2.4 Multi-Robot Systems and Robot-to-Robot (R2R) Communication

Robot-to-Robot (R2R) communication refers to the direct exchange of data, commands, or status information between two or more autonomous robotic systems without immediate human intervention. This form of communication enables coordinated actions, shared situational awareness, and collaborative task execution among robots. R2R communication typically involves peer-to-peer networking protocols, often operates in real-time or near-real-time, and may occur over various physical media (e.g., wireless, wired) depending on the deployment environment.

Robot-to-Robot (R2R) communication connects with concepts such as Machine-to-Machine (M2M) communication, Device-to-Device (D2D) communication, and Vehicle-to-Vehicle (V2V) communication from the autonomous vehicles context. It also relates to SCADA systems, robot swarms, and the deployment of agentic AI in cyber-physical systems.

Communication between entities is fundamental to both cooperation and coordination, playing a central role in networked robotics [28]. Multihop wireless robot networks [29] and multi-robot systems (MRSs) [30] have been extensively studied, emphasizing the importance of reliable and efficient communication protocols.

Swarm robotics, a subset of multi-robot systems, involves collective coordinated behavior emerging from local interaction rules between neighboring robots. This approach fundamentally differs from centralized control systems, as it does not rely on global information or a central control unit.

2.5 Common Protocols and Technologies in Robotic Networking

Several protocols and technologies are commonly used in robotic networking, each with its own strengths and use cases:

- **UDP (User Datagram Protocol):** Often preferred for its low latency and reduced overhead. It is used in various robotic applications, including telesurgery [31] and haptic teleoperation systems.[1]

[1] https://github.com/thomas186/ros_teleop_web

- **WebSockets:** Employed in web-based teleoperation systems for full-duplex communication [32].
- **MQTT (Message Queuing Telemetry Transport):** A lightweight publish-subscribe protocol used in some robotic control architectures [33].

2.6 Challenges in Robotic Networking

Networking in robotics faces several challenges, particularly when operating over the public internet:

- **NAT Traversal and Multicasting:** Many robotic systems operate behind Network Address Translators (NATs), complicating direct peer-to-peer communication and multicasting. NATs can hinder the discovery and communication between robots, especially when using protocols that rely on multicast messaging for discovery and coordination. Overcoming NAT traversal issues is essential for enabling seamless connectivity in distributed robotic systems.
- **Security:** Ensuring secure communication over public networks is essential, especially for critical applications like telesurgery. Robotic systems must protect against eavesdropping, tampering, and unauthorized access, necessitating robust encryption and authentication mechanisms.
- **Scalability:** As robotic systems grow in complexity and number, scalable networking solutions become essential. Handling increased data traffic, managing numerous devices, and maintaining performance require networking architectures that can scale efficiently.
- **Interoperability:** With various protocols and middleware in use, ensuring interoperability between different robotic systems can be challenging. Diverse communication standards can lead to compatibility issues, hindering collaboration among heterogeneous robots.
- **Quality of Service (QoS):** Many robotic applications have strict QoS requirements, such as low latency and high reliability. Meeting these requirements over unpredictable networks like the internet can be difficult, affecting the performance of time-sensitive operations.
- **Bandwidth Limitations:** Limited bandwidth can restrict the transmission of high-volume data such as video streams or sensor data, impacting tasks that rely on rich data exchange.
- **Network Reliability and Latency:** Variable network conditions, including packet loss and latency, can disrupt robotic operations, especially those requiring real-time responses.
- **Mobility Support:** Mobile robots may change network points of attachment, requiring seamless handover and connectivity maintenance without disrupting ongoing communications.

To address these challenges, several solutions are being explored. Virtual Private Networks (VPNs) are sometimes used to securely connect robots across the internet, providing encrypted communication channels, though they may introduce additional latency. Cloud robotics leverages cloud infrastructure for robotic control and data processing, offering scalability and centralized resources. Specialized middleware solutions, such as Zenoh, are being developed to facilitate efficient and secure communication between robotic systems across various network conditions. These middleware platforms aim to abstract the complexity of networking, provide interoperability, and handle issues like NAT traversal and multicasting.

As the field of robotics continues to evolve, networking solutions must adapt to meet the increasing demands for performance, security, and flexibility in robotic communications. Ongoing research and development focus on creating robust, scalable, and secure networking frameworks that can support the complex requirements of modern robotic applications.

3 ROS Communication Stack: Middleware, Protocols, and Internet Connectivity

The Robot Operating System (ROS) [1] has established itself as a fundamental framework in robotics development. This section examines ROS through the lens of networking and communication, aspects central to its evolving role in modern robotics.

ROS conceptualizes a robot as a *network of networks* [34], integrating sensors, actuators, and computational resources. This paradigm enables cohesive environmental response and real-time adaptation.

The communication infrastructure of ROS has evolved significantly from ROS 1 to ROS 2, addressing demands for improved scalability, real-time performance, and security. The adoption of the Data Distribution Service (DDS) as the underlying middleware in ROS 2 represents a substantial change in robotic system communication, enhancing reliability and flexibility across diverse network conditions. Recently, the introduction of Zenoh as an alternative middleware for ROS 2 has further diversified the options for networked communication in robotic systems, offering potential advantages in certain deployment scenarios.

The adoption rate of ROS 2 is increasing rapidly, with usage trends suggesting it may become more prevalent than ROS 1 by 2023 [34]. This shift indicates the robotics community's growing preference for the advanced networking features and communication models offered by ROS 2. However, this transition towards more connected robotic systems brings with it new cybersecurity concerns. A study examining security practices within the ROS community found that nearly three-quarters of respondents felt their efforts to safeguard their robotic systems against cyber threats were inadequate [34]. This finding underscores the emerging security challenges in the increasingly networked domain of robotics and highlights the need for greater attention to cybersecurity measures in robotic system development and deployment.

This section will analyze ROS's communication architecture, its approach to distributed computing, and ongoing efforts to enhance security in networked robotic systems. It will explore how ROS facilitates complex interactions within robotic systems and addresses the challenges of an increasingly connected operational environment.

3.1 ROS 1: XMLRPC and TCPROS/UDPROS

The original Robot Operating System (ROS 1) implemented a communication architecture that was revolutionary for its time in robotics software development. This architecture was built on two primary communication mechanisms: XMLRPC for node discovery and negotiation, and TCPROS/UDPROS for actual data transfer between nodes [35, 36].

3.1.1 XMLRPC for Node Discovery and Negotiation

XMLRPC (XML Remote Procedure Call) served as the backbone for essential functions in ROS 1:

- **Master Node Communication**: For registering and unregistering nodes, topics, and services.
- **Node Discovery**: Allowing nodes to announce their presence and capabilities.

- **Connection Negotiation**: Facilitating communication setup between nodes.
- **Parameter Server**: For storing and retrieving configuration parameters.

While XMLRPC provided flexibility and language independence, it introduced scalability limitations and a single point of failure with the Master node.

3.1.2 TCPROS and UDPROS

Data transfer in ROS 1 primarily utilized two protocols [37, 36]:

- **TCPROS**: The default protocol built on TCP/IP, offering reliable, ordered data delivery.
- **UDPROS**: An alternative UDP-based protocol for faster, but potentially less reliable communication.

Both protocols employed a custom ROS message serialization format for efficient transfer of complex data structures.

3.1.3 Communication Flow in ROS 1

The typical communication flow in ROS 1 involved [36]:

1. Node registration with the Master via XMLRPC.
2. Topic subscription and publisher discovery through the Master.
3. Direct connection negotiation between nodes.
4. Data transfer using TCPROS or UDPROS.

This architecture enabled the creation of flexible and dynamic robotic systems. However, it faced limitations in scalability and performance for large-scale or distributed systems, which ultimately led to the development of ROS 2 with its adoption of DDS as the underlying middleware [35, 3].

3.2 Data Distribution Service (DDS) in ROS 2

The Data Distribution Service (DDS) is the foundation of ROS 2's communication infrastructure and a key component of its security architecture. Understanding DDS is important for improving communication security in ROS 2. This section examines the basic concepts and architecture of DDS, providing context for the subsequent discussion on enhancing ROS 2 security, including the integration of post-quantum cryptography.

3.2.1 ROS 1 to ROS 2 Transition: The Case for DDS

The transition from ROS 1 to ROS 2 marked a significant shift in the underlying communication architecture, with the adoption of DDS as the middleware. This decision was driven by several key factors [35]:

- **End-to-End Middleware:** DDS offered a complete, well-documented solution, reducing the need for maintaining custom code and providing a concrete specification for third-party review and implementation.

- **Technical Credibility:** DDS had a proven track record in mission-critical systems across various industries, lending credibility to its reliability and flexibility.
- **Distributed Discovery:** Unlike ROS 1's centralized master, DDS provides a distributed discovery system, enhancing fault tolerance and flexibility in robotic networks.
- **Quality of Service (QoS):** DDS offers fine-grained control over communication parameters, allowing better adaptation to different network conditions and application requirements.
- **Standardization:** As an OMG standard, DDS ensures interoperability between different implementations and provides a clear specification for future developments.

The DDSI-RTPS (DDS-Interoperability Real Time Publish Subscribe) protocol replaced ROS 1's TCPROS and UDPROS wire protocols for publish/subscribe communications, offering a more robust and standardized approach.

3.2.2 DDS Architecture and Key Features

DDS, developed by the Object Management Group (OMG), is a middleware protocol designed to facilitate data-centric publish-subscribe communication in distributed systems [38]. Its architecture is built around several key components and features [38, 39]:

- **Interoperability:** DDS uses the DDS Interoperability Wire Protocol (DDSI-RTPS) to ensure that applications built on different DDS implementations can communicate effectively [40].
- **Real-Time Publish-Subscribe (RTPS) Protocol:** This forms the backbone of DDS's wire-level communication, designed to meet stringent requirements for discovery, fault tolerance, reliability, and timeliness.
- **Decentralized Architecture:** RTPS operates without central points of failure, enhancing system robustness.
- **Quality of Service (QoS):** Publishers and subscribers interact based on QoS contracts, allowing fine-tuned control over data distribution.
- **Flexible Transport Layer:** DDS allows for different transport mechanisms to suit various network environments and application requirements.

An overview of the DDS architecture is illustrated in Figure 1, showing how these components interact to create a robust, scalable, and efficient data distribution system.

DDS features several technical capabilities that enhance its functionality:

- A built-in discovery service that dynamically identifies and monitors publishers and subscribers without relying on centralized name servers.

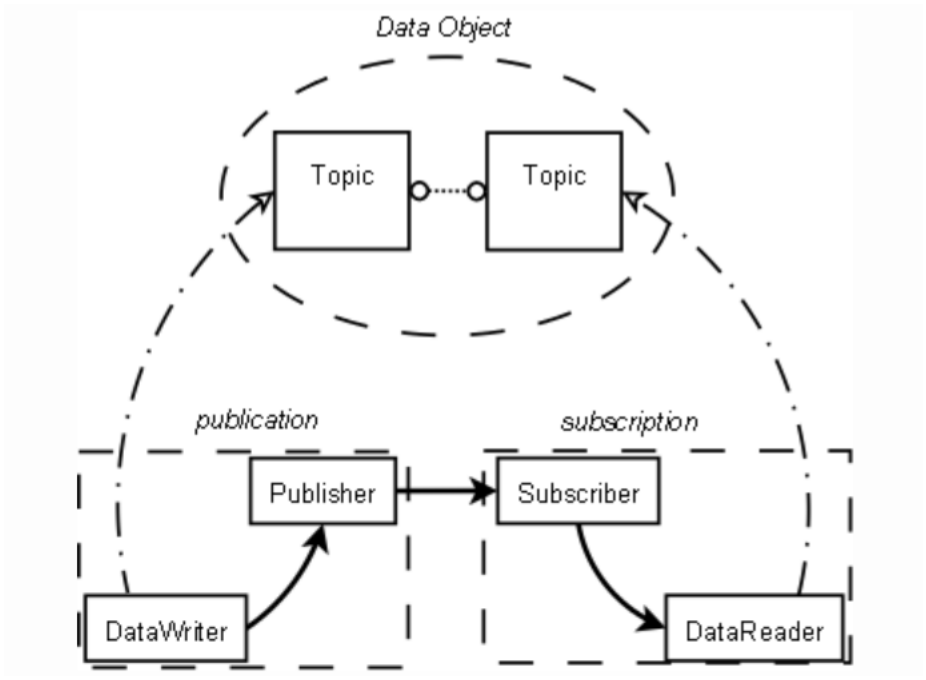

Figure 1: Overview of the DDS architecture, illustrating the data-centric publish-subscribe model and key components (image from OpenDDS).

- A fault-tolerant and decentralized architecture that eliminates single points of failure.
- Extensibility and backward compatibility, allowing for protocol evolution while maintaining interoperability with existing systems.
- Configurable settings to balance reliability and timeliness requirements for each data delivery.
- Scalability to accommodate large-scale networks with potentially thousands of participants.
- Type-safety mechanisms to prevent programming errors from compromising operations on remote nodes.
- Support for various transport protocols to optimize communication based on application needs.

DDS implementations, such as OpenDDS, provide flexibility in the transport layer to accommodate different network conditions and performance requirements. OpenDDS, an open-source implementation of the DDS specification, supports multiple transport protocols [41]:

- **TCP Transport:** Utilizes the Transmission Control Protocol (TCP) for reliable, connection-oriented communication. It is suitable for applications requiring guaranteed delivery.
- **RTPS/UDP Transport:** Uses the Real-Time Publish-Subscribe protocol over the User Datagram Protocol (UDP), enabling interoperability with other DDS implementations and support for real-time communication with configurable reliability.

- **UDP Transport:** Employs unicast UDP for lightweight, low-latency communication without built-in reliability, suitable for scenarios where speed is essential, and occasional data loss is acceptable.
- **Multicast Transport:** Leverages multicast UDP to efficiently distribute data to multiple subscribers simultaneously, reducing network bandwidth usage in one-to-many communication patterns.
- **Shared Memory Transport:** Enables inter-process communication on the same host via shared memory, offering high throughput and low latency by bypassing the network stack.
- **Custom Transports:** Allows developers to implement specialized transport mechanisms tailored to specific application requirements or network environments.

In ROS 2, the integration of DDS is facilitated through the ROS Middleware Interface (RMW), which abstracts the underlying middleware implementation. Figure 2 shows the architecture of ROS 2's internal APIs, highlighting the relationship between the RMW API and the ROS Client Library (RCL) API.

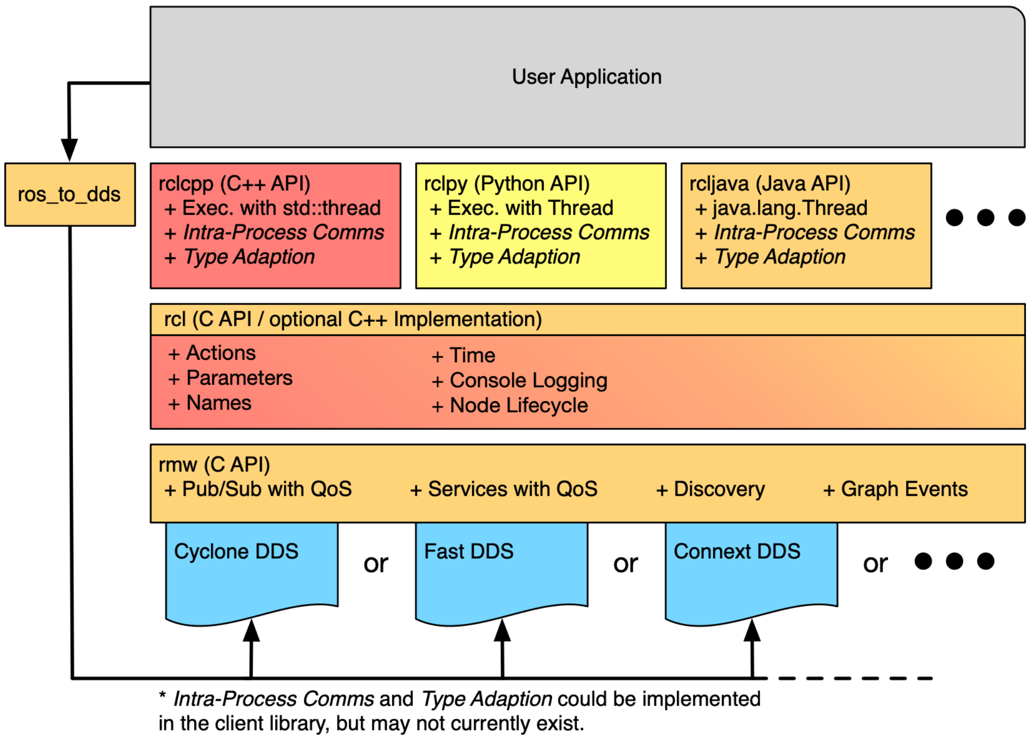

Figure 2: Overview of the DDS architecture, illustrating the data-centric publish-subscribe model and key components (image from OpenDDS [?]).

3.2.3 ROS 2 DDS Middleware Implementations

ROS 2 supports multiple DDS implementations [42], allowing users to choose based on their specific requirements. The main implementations include:

- **eProsima Fast DDS**: The default middleware for most ROS 2 distributions.

- **Eclipse Cyclone DDS**: An alternative implementation gaining popularity in the ROS community.
- **RTI Connext DDS**: A commercial implementation with additional features.
- **OpenDDS**: An open-source implementation of the DDS specification.

These implementations exhibit varying performance characteristics, leading to ongoing evaluations and benchmarking efforts in the ROS community [43, 44, 45]. Comparative studies, such as those between Fast DDS and OpenDDS or Cyclone DDS [46, 45], provide insights into their relative performance in different scenarios.

The choice of DDS implementation can have significant implications for system performance, resource usage, and security features. As we explore enhancing the security of ROS 2 in subsequent sections, particularly in integrating post-quantum cryptographic methods, understanding the characteristics and security capabilities of each implementation will be necessary.

3.3 Zenoh

Zenoh represents a significant development in the domain of robotics middleware, particularly in relation to ROS 2. As an alternative to traditional DDS implementations, Zenoh offers promising solutions to some of the challenges faced in ROS 2 environments. This section explores Zenoh's key features, its performance compared to other messaging systems, and its integration as a ROS 2 middleware. We'll examine how Zenoh addresses discovery overhead issues in ROS 2 and its potential impact on robotics applications. Additionally, we'll discuss Zenoh's security aspects and its selection as an alternate ROS 2 middleware, highlighting its importance in the evolving landscape of robotic communication frameworks.

3.3.1 A Need for an Alternative Middleware?

The adoption of the Data Distribution Service (DDS) as the middleware for ROS 2 brought significant improvements in reliability, scalability, and real-time performance compared to ROS 1. However, as robotic systems have become more complex and distributed, certain limitations of DDS have become apparent, particularly in large-scale and heterogeneous environments [47]. These limitations, along with other factors, have led to a growing need for alternative middleware solutions:

- **Scalability Limitations:** As highlighted in the ROSConDE 2023 presentation [48], one of the primary challenges with DDS is its scalability. The DDS discovery protocol's complexity grows quadratically with the number of participants, topics, readers, and writers in the system:

  $$\mathcal{O}(T \cdot R \cdot W \cdot P^2)\ , \tag{1}$$

  where T is the number of topics, R is the number of readers, W is the number of writers, and P is the number of participants. Such scaling characteristics can lead to significant performance degradation in large-scale robotic deployments.

- **UDP Multicast Dependency:** DDS heavily relies on UDP multicast for discovery, which can be problematic in networks where multicast is not supported or restricted, such as many institutional and large WiFi networks.

- **Large Data Transfer:** DDS can struggle with transferring large data sets efficiently, a common requirement in modern robotics applications involving sensors like high-resolution cameras or LiDAR.
- **Network Performance and Compatibility Issues:** DDS performance can be suboptimal in various network environments, particularly in WiFi networks, which is essential for mobile and field robotics applications. This issue stems from several factors:
  - Modern computing environments are often more suited to TCP-based communications, while DDS primarily relies on UDP [49].
  - DDS's dependency on UDP multicast for discovery can lead to issues in networks where multicast is not supported or restricted.
  - The default UDP kernel and userland buffers are often insufficient for handling large data structures common in modern robotics applications, such as high-resolution sensor data [49].
  - In WiFi networks, DDS performance is highly dependent on network quality and multicast support. If either condition is not met, data delivery can be unreliable [47].
  - These challenges are particularly problematic for ROS 2, which is frequently used in mobile robotics and debugging scenarios where "out of the box" functionality across diverse network environments is essential.
- **Complex Tuning and Suboptimal Out-of-the-Box Experience:** DDS often requires complex parameter tuning to achieve optimal performance, which creates a significant barrier for users without deep middleware expertise. This need for extensive configuration results in a suboptimal out-of-the-box experience, particularly for new users. Moreover, the system is prone to common silent failures that are difficult to diagnose, further complicating the user experience and system setup process [49].
- **Middleware Agnosticism and Abstraction Leakage:** Despite the original goal of creating a middleware-agnostic architecture in ROS 2, the prevalence and dominance of DDS-based implementations have led to DDS-specific details permeating through various layers of the ROS 2 stack, including the RMW, rcl, and rclcpp layers. This leakage through the RMW interface compromises middleware abstraction, making it challenging to implement and integrate non-DDS middleware solutions seamlessly [49].
- **Complexity in Containerized Environments:** The setup and configuration of DDS in containerized environments have proven to be complex [49], posing significant challenges to developers and system integrators.

These limitations have led the robotics community to explore alternative middleware solutions. In 2023, Open Robotics conducted a comprehensive study on ROS 2 RMW alternatives [47], evaluating various middleware options against a set of defined requirements. The ideal alternative would offer improved performance in diverse network environments, simplify deployment and configuration, provide better handling of large data transfers, and adhere more strictly to the principle of middleware agnosticism originally envisioned for ROS 2.

The study identified several key requirements for an alternative middleware, including:

- Efficient discovery mechanisms to reduce overhead in large-scale systems
- Better support for heterogeneous networks, including those with limited multicast capabilities
- Improved performance in transferring large data sets and in challenging network conditions
- Simplified configuration and use, reducing the need for complex tuning
- Support for peer-to-peer, routed, and brokered communication patterns
- Ability to handle data in motion, data at rest, and distributed computations within a single framework

As a result of this evaluation, Zenoh was selected as the first non-DDS protocol to be natively supported in ROS 2. Zenoh addresses many of the limitations of DDS and meets most of the identified requirements. It offers a more efficient discovery mechanism, better support for heterogeneous networks, improved performance in challenging network conditions, and aims for simplicity in configuration and use.

The selection of Zenoh as an alternative middleware for ROS 2 represents a significant shift in the robotics community's approach to communication middleware. It reflects the evolving needs of robotic systems, particularly in scenarios involving large-scale deployments, heterogeneous networks, and resource-constrained devices.

This transition to supporting alternative middlewares like Zenoh in ROS 2 opens new possibilities for robotics applications. It potentially allows for more efficient communication in diverse environments, from cloud-based robot fleets to swarm robotics applications, and could facilitate better integration of robotics with emerging IoT and edge computing paradigms.

3.3.2 Zenoh Overview

Zenoh is a communication protocol designed to meet the evolving needs of modern robotics and distributed systems by providing a unified framework for managing data in motion, data at rest, and computations. It integrates publish/subscribe and query paradigms, supporting various communication models including peer-to-peer, routed, and brokered topologies [50].

One of Zenoh's key features is its flexibility in network configurations, which addresses scalability issues inherent in traditional DDS implementations. Unlike DDS, which relies on multicast UDP for discovery and can lead to quadratic scaling problems in large networks, Zenoh can use a unicast-based discovery mechanism using a gossip protocol [50]. This approach reduces discovery overhead by allowing each node to communicate its presence to a single entity, avoiding the need for widespread broadcasting.

The Zenoh router is required only during the startup of ROS nodes to facilitate initial discovery and interconnectivity. Once nodes establish peer-to-peer communication, the router is no longer a single point of failure and can be restarted or reconfigured without disrupting ongoing communications. This design contrasts with ROS 1, where the ROS master was essential for maintaining communication channels.

Zenoh is engineered for high throughput and low latency, making it suitable for resource-constrained environments common in robotics. Its architecture features minimal wire overhead (4–6 bytes) and supports communication locality, which reduces the energy footprint of data transmission.

Additional features of Zenoh include downsampling capabilities, allowing users to reduce data rates by sampling messages at lower frequencies. This flexibility aids in managing bandwidth and processing resources both across communication bridges and within individual robots [50]. Zenoh also provides access control mechanisms, such as allow or deny lists, enabling fine-grained control over data flows and enhancing system security.

Zenoh has been adopted in various applications, including robotics, autonomous vehicles, internet gaming, and telecommunications. In robotics, it facilitates robot-to-robot communication, internet-scale monitoring, real-time teleoperation, and vehicle-to-anything (V2X) communication, as demonstrated by its use in initiatives like CARMA and the Indy Autonomous Challenge [50].

3.3.3 zenoh-bridge-ros2dds: Zenoh Bridge for ROS 2 over DDS

While Zenoh can serve as a complete middleware replacement for DDS in ROS 2, an alternative approach involves using the `zenoh-bridge-ros2dds` [51] [52], which bridges ROS 2 communication over DDS using Zenoh without replacing the underlying middleware. This bridge allows existing ROS 2 applications to benefit from Zenoh's networking capabilities, particularly in challenging network environments.

The *Zenoh Bridge for ROS 2 over DDS* serves as an intermediary solution that enhances ROS 2 communications by leveraging Zenoh's efficient networking capabilities without entirely replacing the underlying DDS middleware. This bridge addresses several networking challenges inherent in traditional DDS implementations, thereby improving the scalability and reliability of robotic communication frameworks.

It is available in two primary forms:

- **zenoh-plugin-ros2dds**: A dynamic library loaded by a Zenoh router, integrating ROS 2 communications within an existing Zenoh infrastructure.
- **zenoh-bridge-ros2dds**: A standalone executable that facilitates the bridging process independently of the Zenoh router, suitable for decentralized deployments or resource-constrained environments.

Both forms share identical features and configurations, offering flexibility in deployment based on specific network architectures and requirements.

The `zenoh-bridge-ros2dds` operates by intercepting all ROS 2 communications that utilize DDS and routing them over Zenoh. This approach offers several advantages:

- **Enhanced Network Performance**: By leveraging Zenoh's efficient networking protocols, the bridge improves communication over networks with limited multicast support, such as WiFi or large institutional networks [49].
- **Reduced Discovery Overhead**: The bridge mitigates the quadratic scaling issues of DDS discovery by utilizing Zenoh's gossip-based discovery mechanism. This approach significantly reduces the amount of discovery traffic, enhancing scalability in large systems [49].
- **Seamless Integration**: Existing ROS 2 nodes and tools can operate without modification, as the bridge transparently handles communication over Zenoh while maintaining compatibility with DDS-based nodes.
- **Fault Tolerance**: The Zenoh router used by the bridge is not a single point of failure. Once nodes are interconnected in peer-to-peer mode, the router can be restarted or reconfigured without disrupting ongoing communications [49].
- **Flexible Deployment**: The bridge supports various network configurations, making it suitable for multi-robot systems and remote monitoring applications where network conditions are unpredictable or suboptimal.

Using the `zenoh-bridge-ros2dds` provides operational benefits without requiring significant changes to existing ROS 2 systems:

- **Maintains DDS Features Locally**: Local ROS 2 nodes continue to use DDS and its Quality of Service (QoS) settings, preserving real-time capabilities and other DDS-specific features within the robot or system.
- **Transparent to ROS 2 Tools**: All ROS 2 topics, services, and actions are visible across bridges, ensuring compatibility with tools like `rviz2` and `ros2` command-line utilities.
- **Namespace Configuration**: The bridge allows setting a ROS namespace at the bridge level, simplifying multi-robot deployments by avoiding the need to configure namespaces on each node individually.
- **Access Control**: Zenoh's access control mechanisms enable fine-grained control over data flows, allowing for the implementation of allow or deny lists for enhanced security.

The `zenoh-bridge-ros2dds` serves as a solution for enhancing ROS 2 communication over DDS by leveraging Zenoh's efficient networking capabilities. It addresses key challenges associated with DDS in complex or constrained network environments without necessitating significant alterations to existing ROS 2 applications. By focusing on networking improvements and maintaining DDS compatibility, the bridge facilitates more robust and scalable robotic systems suited for modern distributed applications.

3.3.4 rmw_zenoh: Zenoh ROS 2 middleware

An alternative to using Zenoh as a bridge is to adopt it as a full ROS 2 middleware through the `rmw_zenoh` implementation [53]. This approach replaces DDS entirely, allowing ROS 2 nodes to communicate directly over Zenoh.

The `rmw_zenoh` implementation uses the C bindings for Zenoh, ensuring compatibility and performance [49]. Although Zenoh is implemented in Rust, this detail is completely hidden behind the RMW (ROS Middleware) layer, providing a seamless integration with ROS 2's client libraries.

Key features and considerations of `rmw_zenoh` include:

- **Full Middleware Replacement**: By replacing DDS, `rmw_zenoh` allows all nodes to communicate over Zenoh, without the need for DDS dependencies.
- **Visibility of Nodes**: With `rmw_zenoh`, all nodes are visible everywhere in the network, enhancing discovery and communication [49].
- **Service Communication**: The implementation relies on Zenoh's query mechanisms to enable service communications between nodes [49].
- **Data Handling**: Each node caches its publications, similar to DDS, avoiding the potential overwrites that can occur with bridges caching data from multiple publishers on the same topic [49].
- **Downsampling Capabilities**: `rmw_zenoh` supports downsampling within the robot, offering more generalized and flexible control over data rates compared to the bridge [49].

There has been discussion about whether to implement the RMW API directly in Rust or use the C/C++ bindings. Some suggest that using Rust directly could be beneficial in terms of performance and development time [47].

The adoption of `rmw_zenoh` represents a more profound shift from DDS, potentially offering greater performance improvements and simplifications in configuration and deployment. However, it also requires thorough testing and validation to ensure compatibility and stability within the ROS 2 ecosystem.

3.3.5 Comparison with DDS

The study [54] evaluated the performance of three ROS 2 middlewares (FastRTPS, CycloneDDS, and Zenoh) over a mesh network with a dynamic topology, focusing on a scenario of exploring extreme extra-terrestrial environments using a multi-robot system (MRS).

Zenoh emerged as the most promising solution for using ROS 2 on a mesh network in this scenario, particularly excelling in:

- Reduced delay
- Better reachability
- Lower CPU usage
- Competitive performance on data overhead and RAM usage

Key findings for Zenoh compared to the DDS implementations:

- Reduced delay by 76% compared to FastRTPS and 69.86% compared to CycloneDDS

- Increased reachability by 146.93% compared to FastRTPS and 58.17% compared to CycloneDDS
- Reduced CPU usage by 41.27% compared to FastRTPS and 39.76% compared to CycloneDDS
- Slightly higher RAM usage (60.50% increase vs FastRTPS, 86.03% increase vs CycloneDDS)
- Mixed results on data overhead (4.36% higher than FastRTPS, 48.14% higher than CycloneDDS)

Zenoh performed particularly well for small and medium-sized messages, while its advantages were less pronounced for larger messages.

The study emphasized that the choice of middleware should depend on the specific requirements of the mission. For scenarios prioritizing power consumption and network reachability, Zenoh appears to be the best choice. However, if data throughput is the primary concern, CycloneDDS might be more suitable. The researchers noted that Zenoh's performance with very large messages (beyond 64 KB) needs further investigation.

The study highlighted the importance of considering factors like reachability, data overhead, and CPU usage in extra-terrestrial exploration scenarios, where power consumption and maintaining consistent connectivity are important.

These conclusions provide a comparison of the ROS 2 middlewares, with a particular focus on their performance in challenging, dynamic network environments relevant to space exploration and multi-robot systems.

3.4 Communicating ROS 2 Over the Internet

As robotic systems become increasingly distributed and interconnected, the need to extend ROS communication beyond local networks has become essential. This section explores the challenges and solutions for enabling ROS to operate effectively over the Internet, focusing on key networking issues and the various approaches developed to address them.

3.4.1 Challenges in Internet-based ROS Communication

Two significant challenges in extending ROS communication over the Internet are the lack of multicast support and the prevalence of Network Address Translation (NAT).

Multicasting Limitations Multicast, a method for efficient group communication, is not widely supported on the public Internet, which poses problems for ROS's discovery mechanisms.

Network Address Translation (NAT) Issues NAT, while essential for IPv4 address conservation and network security, complicates direct peer-to-peer connections often required in robotics applications.

To overcome these challenges, several strategies have been developed, including the use of relay servers, ICE (Interactive Connectivity Establishment) protocols, VPN solutions, application-layer protocol adaptations, and cloud-based architectures. These approaches aim to facilitate ROS communication across diverse network environments, balancing factors such as latency, scalability, and ease of deployment.

3.4.2 Solutions for Internet-enabled ROS Communication

ROS Bridges *rosbridge* [55] implements a JSON-based protocol that facilitates communication between ROS 2 and ROS 1 systems, as well as web clients. This protocol extends ROS functionality to web-based applications through a standardized interface, enabling the development of web applications that can interact with ROS systems without direct ROS implementation.

The rosbridge suite consists of three main components: a library for interpreting JSON-formatted strings and executing ROS operations, a server that exposes this functionality, and a service provider for system meta-information. It supports multiple transport layers, primarily WebSockets and TCP, allowing for adaptation to various network environments.

Key features of the rosbridge protocol include support for topic, service, and action operations, message fragmentation for large data transmission, compression for efficient data handling, and configurable status messaging. These features enable robust and flexible communication between ROS and web-based systems.

Manzi et al. [32] demonstrate the effectiveness of this approach through a cloud-based system that enables real-time, full-duplex communication between a ROS-based robot and a control platform. Their architecture, utilizing Robot Web Tools and a rosbridge server, showcases how this technology can address network accessibility challenges and facilitate secure robot connections to cloud platforms without direct network integration.

By providing a standardized interface, rosbridge simplifies the integration of web technologies with ROS-based robotic systems across various network configurations, making it a valuable tool for internet-based robotic applications.

RTPS Relay Another approach to enable ROS communication over the Internet is the use of RTPS Relays. The RTPS Relay is a service designed to forward RTPS messages between participants, facilitating communication even when they are behind NAT firewalls or in environments without multicast support [56, 57]. By acting as an intermediary, the RTPS Relay allows participants to discover each other and exchange messages over unicast UDP, which is generally supported across networks and compatible with NAT devices.

The RTPS Relay operates by intercepting RTPS datagrams from participants and forwarding them to other relays or participants based on an association table. This mechanism addresses the limitations of multicast-dependent discovery protocols in DDS by providing a unicast-based solution that can traverse NAT boundaries. In cloud deployments where multicast is unavailable, the RTPS Relay solves the bootstrapping problem by serving as a well-known point for discovery and communication.

While the RTPS Relay enhances connectivity in distributed systems, it introduces additional infrastructure requirements and may impact latency and scalability depending on network conditions and relay configurations. Careful deployment and potential load balancing strategies are necessary to ensure that the relay does not become a bottleneck or single point of failure in the communication architecture.

Interactive Connectivity Establishment (ICE) Interactive Connectivity Establishment (ICE) is a protocol adapted for use in DDS implementations to enable direct peer-to-peer network connections between hosts separated by NAT firewalls [56, 57]. ICE serves as an optimization technique in scenarios where relay servers would otherwise be necessary.

The core concept of ICE involves discovering and exchanging potential connection points between peers, then attempting to establish direct connections. In DDS systems, ICE can be applied to various types of endpoints, enhancing connectivity in distributed robotic systems deployed across diverse network environments.

By implementing ICE, DDS systems can overcome NAT-related communication barriers, enabling more efficient peer-to-peer connections and reducing reliance on relay servers. This approach can improve latency and scalability by minimizing the need for intermediary services in the communication path.

Zenoh As mentioned before in Sections 3.3.3 and 3.3.4, Zenoh offers two main approaches for enhancing ROS 2 communication over the internet:

- **zenoh-bridge-ros2dds**: A bridge that routes ROS 2 communications over Zenoh without replacing the underlying DDS middleware.
- **rmw_zenoh**: A full Zenoh implementation as an alternative ROS 2 middleware.

Both approaches aim to improve discovery mechanisms, reduce overhead, and enhance performance in challenging network environments. Zenoh's efficient protocol and flexible architecture make it particularly suitable for large-scale robotic deployments and scenarios involving heterogeneous networks. By using Zenoh, ROS 2 systems can potentially overcome many of the networking limitations associated with traditional DDS implementations.

WebRTC WebRTC (Web Real-Time Communication) is a technology that enables direct peer-to-peer communication between web browsers or applications without the need for plugins or additional software installations. It has been successfully applied in robotics for various purposes, including video streaming from robots and implementing teleoperations for ROS robots. For ROS 1, the webrtc_ros project [58] provides a solution for streaming ROS image topics using WebRTC, demonstrating the potential of this technology in robotic applications. While initially developed for ROS 1, similar approaches can be adapted for ROS 2 to enable efficient, low-latency communication over the internet.

Virtual Private Networks (VPNs) VPNs offer a secure method for connecting ROS systems over the Internet by creating an encrypted tunnel between networks. This approach can be particularly useful for security-critical applications in robotics [24].

4 Security in ROS 2

4.1 Overview of Security Considerations in Robotics

As robotics systems become increasingly ubiquitous and integral to various industries, the need to incorporate robust cybersecurity measures has never been more important. The rapid deployment of robotic technologies often prioritizes functionality and time-to-market, inadvertently sidelining essential security mechanisms during development, deployment, and operational phases [59]. This oversight renders contemporary robotic systems susceptible to cyber-attacks, compromising not only data integrity and privacy but also the physical safety of environments where these robots operate. Cyber-physical systems, which merge computational processes with physical actions, present unique security challenges that necessitate comprehensive protection strategies from the earliest design stages [60]. Moreover, the reliance on communication protocols such as the Data Distribution Service (DDS) within frameworks like ROS 2 introduces additional vulnerabilities, particularly in internet-enabled deployments where multicast and network address translation (NAT) can complicate secure communications [61]. The advent of post-quantum cryptography emphasizes the need for quantum-resistant encryption schemes to protect command and control mechanisms in mobile and autonomous robotic systems [62]. Addressing these security considerations is essential to ensure the safe integration of robotic technologies in the economy.

4.2 Post-Quantum Cryptography

Post-Quantum Cryptography (PQC), also referred to as quantum-resistant cryptography, is an emerging field aimed at developing cryptographic systems that can withstand attacks from both classical and quantum computers. The standardization process for PQC is ongoing, with algorithms focusing on two main areas: Key Encapsulation Mechanisms (KEM) for secure key exchange and encryption [63], and digital signature schemes [64].

Current PQC algorithms are primarily based on five mathematical approaches: lattices, multivariate polynomials, error-correcting and error-detecting codes, hash-based signatures, and isogenies of elliptic curves. Notable progress has been made in standardization efforts, with NIST approving several algorithms. Kyber [65], a lattice-based KEM, and Dilithium [66], a lattice-based signature scheme, have been standardized. Additionally, SPHINCS+ [67], a stateless hash-based signature algorithm, has received approval.

These post-quantum algorithms exhibit significantly different characteristics compared to traditional cryptographic methods, particularly in terms of key sizes and the length of encrypted or signed data. As the field evolves, new algorithms may be standardized, necessitating a flexible approach to implementation. The ultimate goal is to develop crypto-agile APIs that allow seamless integration of various PQC algorithms into applications, regardless of the size of cryptographic materials, ensuring long-term security in the face of advancing quantum computing capabilities.

The integration of PQC into established security protocols is an important step in preparing systems. A relevant example of this integration process can be seen in the efforts to incorporate PQC into Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). These efforts serve as valuable case studies, offering insights that can be applied to other security protocols, particularly in the context of DDS security (given its similarity).

Recent research highlights the importance of introducing PQC into robotic systems [62].

4.2.1 Integrating Post-Quantum Cryptography in TLS/DTLS

The integration of PQC into established security protocols like Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) is an important step in preparing for the quantum era. TLS 1.2 [68] and DTLS 1.2 [69] utilize a handshake process for peer authentication, algorithm negotiation, and shared secret computation. These protocols rely on X.509 certificates, cipher suites, and key exchange methods such as Diffie-Hellman. The handshake involves a series of exchanges, including ClientHello/ServerHello messages with nonces, certificate and signature exchanges, and Diffie-Hellman-based key derivation using HMAC.

Significant advancements have been made in the more recent TLS 1.3 [70] and DTLS 1.3 [71] specifications. These versions enhance both security and performance by employing only modern, secure cryptographic algorithms and streamlining the handshake process. TLS 1.3 introduces improvements such as Elliptic Curve Diffie-Hellman (ECDHE) for forward-secret key exchange, ChaCha20-Poly1305 for authenticated encryption, and an updated key derivation process based on the HKDF scheme.

The integration of post-quantum primitives into the TLS handshake process has been the subject of several studies [72, 73, 74]. Proposals include replacing RSA/ECDSA signatures with post-quantum alternatives and substituting Diffie-Hellman key exchange with KEM operations. Bos et al. [72] demonstrated the feasibility of replacing Diffie-Hellman with KEM in TLS 1.2 and provided a security proof for this approach. For TLS 1.3, a hybrid approach has been proposed [75], combining post-quantum algorithms with traditional schemes like ECDH. This hybrid model offers quantum resistance while maintaining pre-quantum security assurances during the transition period.

Recent years have seen significant progress in implementing PQC in widely-used cryptographic libraries. The Open Quantum Safe project has developed a fork of OpenSSL [76] that incorporates quantum-resistant algorithms for key exchange and digital signatures in TLS 1.2 and 1.3. This implementation utilizes the liboqs library [77], which provides a comprehensive collection of post-quantum cryptographic algorithms. Furthermore, the development of oqs-provider [78], an OpenSSL 3 provider, has expanded the availability of quantum-resistant algorithms in the OpenSSL ecosystem. In the realm of datagram-based communication, wolfSSL has introduced support for post-quantum cryptography in DTLS 1.3 [79], enabling the use of quantum-resistant algorithms in DTLS1.3.

4.3 ROS 2 / DDS Security

4.3.1 Overview

The evolution of security measures in the Data Distribution Service (DDS) reflects the growing importance of robust cybersecurity in distributed systems. Initially, DDS security relied on Transport Layer Security (TLS) or Datagram TLS (DTLS) protocols to ensure data integrity and confidentiality, primarily due to the absence of a dedicated security framework within the DDS standards [80, 81].

Various DDS-compliant products incorporated security mechanisms based on these protocols. For instance, RTI DDS employed DTLS for notification encapsulation, while OpenSplice DDS utilized domain partitioning for access control. eProsima Fast DDS and RTI Connext extended support to secure TCP transports with TLS [82, 83]. However, the implementation of TLS was not standardized across all DDS implementations [84].

These initial approaches had limitations, particularly in supporting multicast communications due to DTLS's inherent client/server structure. This shortcoming highlighted the need for a more comprehensive DDS security specification. Soroush et al. conducted a comparative study of DDS security implemented over secure TLS/DTLS transports against RTI's beta version of the DDS Security Specification, further emphasizing this need [85].

In response to these challenges, the DDS-Security specification was developed [86]. This specification extends the core DDS standard by introducing predefined security features through a Service Plugin Interface (SPI) framework. The DDS-Security model is designed to work over any transport protocol while maintaining configurable Quality of Service (QoS) settings, which are essential for real-time and mission-critical applications. It also offers interoperability across vendor implementations.

A standout feature of DDS-Security is its implementation of mutual authentication, where both the publisher and subscriber authenticate each other. This bidirectional verification enhances security by ensuring that all communicating parties are trusted and authorized, a significant advancement over many traditional security models.

The DDS-Security specification defines five key security plugins:

1. **Authentication:** This plugin, `DDS:Auth:PKI-DH`, verifies the identity of domain participants using a trusted Certificate Authority (CA). It employs RSA or ECDSA signature algorithms (with 2048-bit and 256-bit NIST P-256 curve key sizes, respectively) and uses DHE or ECDHE for key exchange in a 3-message handshake protocol.

2. **Cryptography:** The `DDS:Crypto:AES-GCM-GMAC` plugin manages encryption, signing, and hashing operations. It ensures data confidentiality and integrity using AES in Galois Counter Mode (AES-GCM) and Galois Message Authentication Code (AES-GMAC).

3. **Access Control:** This component regulates permissions for DDS operations, allowing for fine-grained control over different applications within a DDS domain.

4. **Logging:** While not universally implemented, this plugin facilitates the auditing of security-related events, enhancing the ability to monitor and respond to potential security incidents.

5. **Data Tagging:** This optional feature allows for the addition of metadata tags to data samples, providing an extra layer of information management.

These security features address several potential vulnerabilities in DDS systems, including unauthorized subscriptions and publications, data tampering, replay attacks, and unauthorized data access. The design of these plugins was driven by key requirements such as scalable performance, system robustness, ease of use, and compatibility with existing security infrastructures, while maintaining the data-centric nature of DDS and avoiding the introduction of centralized components that could become single points of failure.

The DDS-Security model offers significant advantages over the earlier TLS-based approaches. Notably, it provides mutual authentication, requiring both communicating parties to authenticate each other, and supports secure multicast communication, enabling scalable one-to-many communication—a feature that TLS does not inherently support. The pluggable nature of the Security Plugins also allows for the integration of custom encryption or signing algorithms, offering flexibility beyond the fixed set of algorithms provided by TLS [84].

The introduction of this standardized security framework marks a significant advancement in the security capabilities of DDS, providing a more uniform and comprehensive approach to securing DDS-based communications across different vendor implementations.

4.3.2 Authentication Process in DDS Security

The authentication process in DDS security is an important component that ensures the identity and trustworthiness of participants in a DDS domain. This process is particularly significant in the context of encrypted communication and the integration of post-quantum cryptography. The authentication workflow can be summarized in the following high-level steps:

1. **Discovery:** Participants discover each other through the DDS discovery protocol, exchanging initial identity and permissions tokens.

2. **Identity Validation:** Each participant validates the other's identity using the received tokens.

3. **Handshake Initiation:** One participant initiates a handshake request, generating a message token.

4. **Handshake Exchange:** Participants exchange a series of handshake messages, each containing cryptographic material for authentication.

5. **Authentication Completion:** Both participants process the received messages, verifying the authenticity of the other party.

6. **Shared Secret Establishment:** Upon successful authentication, a shared secret is established between the participants.

7. **Permission Verification:** Participants retrieve and verify each other's permissions credentials.

This process ensures mutual authentication, where both the publisher and subscriber verify each other's identity. This bidirectional authentication is a key feature that distinguishes DDS security from many traditional security models, such as those commonly used in HTTPS where typically only the server authenticates to the client.

Figure 3 illustrates a simplified version of this authentication workflow. The process involves multiple message exchanges, ensuring that both participants can verify each other's credentials before establishing a secure communication channel.

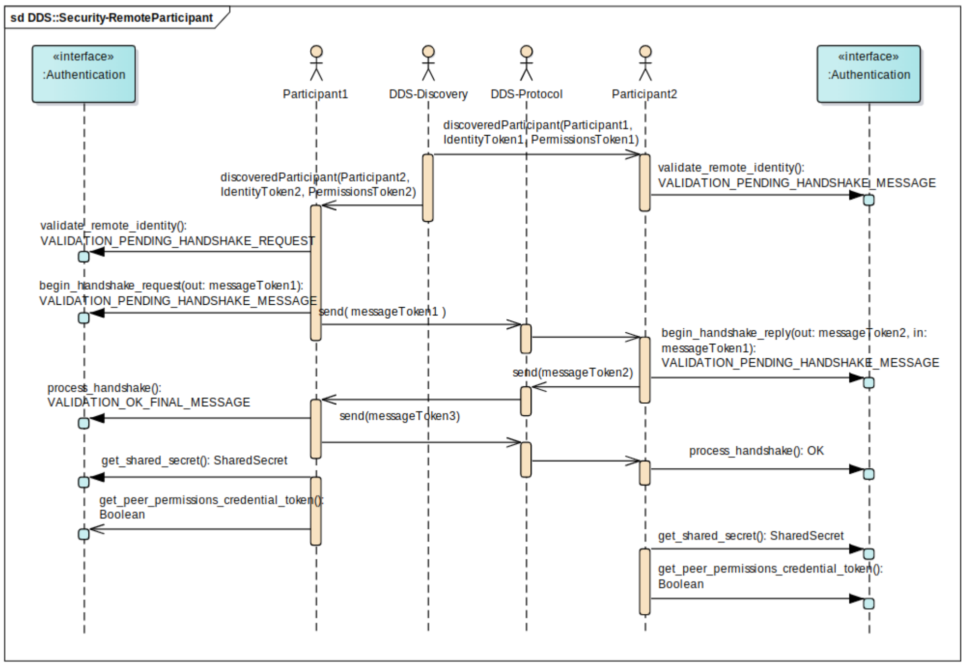
sd DDS::Security-RemoteParticipant
«interface» :Authentication
Participant1
DDS-Discovery
DDS-Protocol
Participant2
«interface» :Authentication
discoveredParticipant(Participant1, IdentityToken1, PermissionsToken1)
validate_remote_identity(): VALIDATION_PENDING_HANDSHAKE_MESSAGE
discoveredParticipant(Participant2, IdentityToken2, PermissionsToken2)
validate_remote_identity(): VALIDATION_PENDING_HANDSHAKE_REQUEST
begin_handshake_request(out: messageToken1): VALIDATION_PENDING_HANDSHAKE_MESSAGE
send(messageToken1)
begin_handshake_reply(out: messageToken2, in: messageToken1): VALIDATION_PENDING_HANDSHAKE_MESSAGE
send(messageToken2)
process_handshake(): VALIDATION_OK_FINAL_MESSAGE
send(messageToken3)
process_handshake(): OK
get_shared_secret(): SharedSecret
get_peer_permissions_credential_token(): Boolean
get_shared_secret(): SharedSecret
get_peer_permissions_credential_token(): Boolean

Figure 3: DDS authentication workflow (image taken from DDS Security Specification).

4.3.3 Comparison with TLS (HTTPS)

While both DDS Security and TLS aim to secure communications, their approaches to authentication and communication models differ significantly:

- **Mutual vs. One-Way Authentication:**
  - **DDS Security:** Implements mutual authentication, requiring both communicating parties (publishers and subscribers) to present and verify each other's certificates. This ensures that both entities are trusted and authorized.
  - **TLS (HTTPS):** Typically implements one-way authentication, where only the server presents a certificate to authenticate itself to the client. The client does not present a certificate unless client authentication is specifically configured.
- **Communication Model:**
  - **DDS Security:** Designed for multicast and publish/subscribe communication patterns, making it suitable for distributed and real-time systems like robotics.
  - **TLS (HTTPS):** Primarily designed for unicast client-server communication, which is more suitable for web browsing and similar applications.
- **Security Plugins and Flexibility:**
  - **DDS Security:** Utilizes a pluggable architecture through Security Plugins, allowing for the integration of custom encryption or signing algorithms, including post-quantum cryptographic methods.

  - **TLS (HTTPS):** Offers a fixed set of cryptographic algorithms, with limited flexibility for custom integrations.

- **Key Exchange Mechanism:**
  - **DDS Security:** Often employs Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) for key exchange, supporting both traditional and post-quantum cryptographic algorithms.
  - **TLS (HTTPS):** Primarily uses DH, ECDH, or RSA for key exchange, with limited support for post-quantum algorithms.

These differences highlight DDS Security's suitability for complex, distributed systems requiring robust, two-way authentication and flexible security configurations, which are essential in environments like robotics and IIoT.

4.3.4 Post-Quantum Cryptography: PQSec-DDS

The integration of Post-Quantum Cryptography (PQC) into the Data Distribution Service (DDS) represents a significant advancement in securing robotic systems and Industrial Internet of Things (IIoT) applications against future quantum threats. Our previous work, PQSec-DDS [87], is one of the pioneering efforts in this domain, addressing the need for quantum-resistant security in DDS-based communications.

PQSec-DDS leverages the DDS Security Specification's Service Plugin Interface (SPI) framework to integrate custom security plugins, focusing primarily on the Authentication plugin. This approach allows for the seamless incorporation of post-quantum cryptographic algorithms into the existing DDS security architecture.

The authentication process (shown in Figure 4) has been adapted to incorporate quantum-resistant algorithms. Specifically:

- The handshake messages now include post-quantum key encapsulation mechanisms (KEMs) for key exchange.
- Digital signatures used in the authentication process are replaced with post-quantum signature schemes.
- The shared secret establishment phase now uses quantum-resistant methods to derive encryption keys.

These modifications ensure that the authentication process remains secure even in the face of potential quantum computing threats, while maintaining the core principles of mutual authentication and secure key establishment that are central to DDS security.

The successful completion of this authentication process enables subsequent encrypted communication between DDS participants. It ensures that not only is the data protected from eavesdropping through encryption, but also that the identities of the communicating parties are verified, preventing unauthorized data injection or access.

Key aspects of PQSec-DDS include:

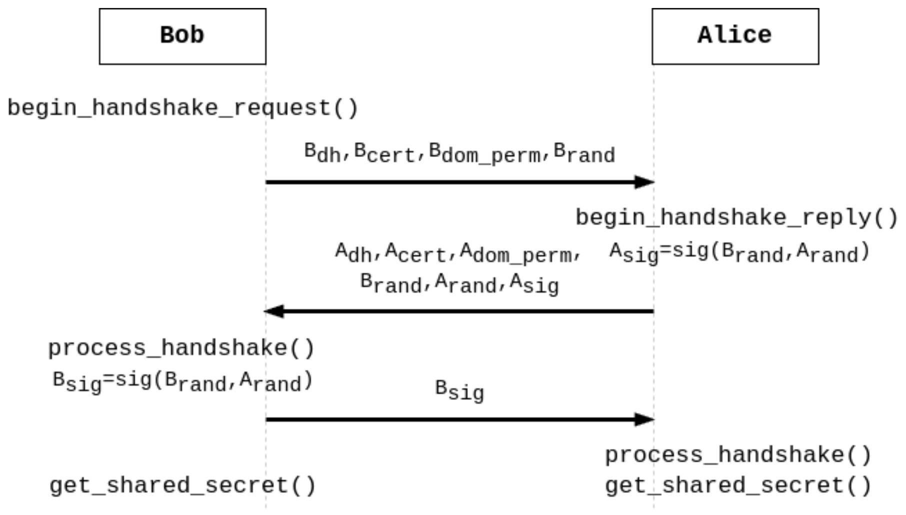
Bob
Alice
begin_handshake_request()
$B_{dh}, B_{cert}, B_{dom\_perm}, B_{rand}$
begin_handshake_reply()
A_{sig}=sig(B_{rand}, A_{rand})
$A_{dh}, A_{cert}, A_{dom\_perm}, B_{rand}, A_{rand}, A_{sig}$
process_handshake()
B_{sig}=sig(B_{rand}, A_{rand})
B_{sig}
process_handshake()
get_shared_secret()
get_shared_secret()

Figure 4: DDS Authentication handshake with mutual authentication between two domain participants, Alice and Bob. The discovery mechanism of the participants is omitted for clarity.

- **Crypto-Agility:** The plugin design facilitates the integration of various post-quantum algorithms, allowing for flexibility and future-proofing against evolving cryptographic standards.

- **Authentication Handshake Modification:** PQSec-DDS adapts the DDS authentication handshake to use post-quantum key encapsulation mechanisms (KEMs) and signature schemes, as illustrated in Figure 5.

- **Library Integration:** The implementation utilizes the liboqs library and the Open Quantum Safe's oqs-provider, enabling the use of standardized post-quantum cryptographic primitives.

Figure 5 depicts the integration of post-quantum KEMs and signatures into the DDS Authentication handshake. The blue highlights indicate the key changes required to transition from classical to post-quantum cryptography.

The PQSec-DDS plugin demonstrates the feasibility of integrating PQC into DDS, paving the way for quantum-resistant secure communication in robotic and IIoT systems. However, it's important to note that this field is still in its early stages, and further research is needed to fully understand the performance implications and practical challenges of deploying PQC in real-world DDS applications.

Further research is needed to fully understand the performance implications and practical challenges of deploying PQC in real-world DDS applications. Future work should focus on:

- Integrate PQC into different DDS implementations.

- Benchmarking the integration of PQC into Cyclone DDS and with different DDS vendors.

- Conducting network benchmarks with a focus on scalability, particularly with a large number of publishers and subscribers.

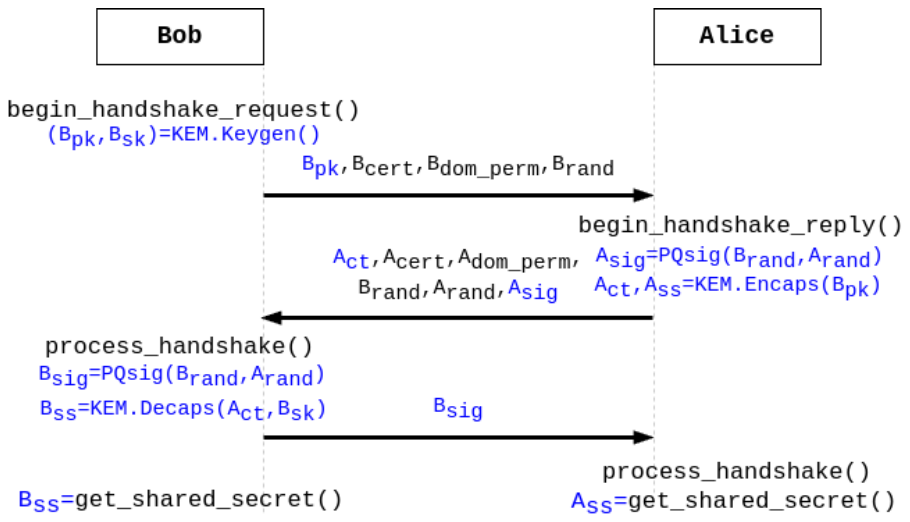
Bob
Alice
begin_handshake_request()
(B_{pk}, B_{sk})=KEM.Keygen()
$B_{pk}, B_{cert}, B_{dom\_perm}, B_{rand}$
begin_handshake_reply()
$A_{ct}, A_{cert}, A_{dom\_perm}, B_{rand}, A_{rand}, A_{sig}$
A_{sig}=PQsig(B_{rand}, A_{rand})
A_{ct}, A_{ss}=KEM.Encaps(B_{pk})
process_handshake()
B_{sig}=PQsig(B_{rand}, A_{rand})
B_{ss}=KEM.Decaps(A_{ct}, B_{sk})
B_{sig}
process_handshake()
B_{ss}=get_shared_secret()
A_{ss}=get_shared_secret()

Figure 5: Integration of Post-Quantum KEMs and signatures into DDS Authentication handshake with mutual authentication. The discovery mechanism of domain participants is omitted for clarity. The required changes are highlighted in blue.

- Comparing performance with alternative ROS2 middlewares to provide a comprehensive understanding of the impact of PQC on different DDS implementations.

4.3.5 Comparison with TLS (HTTPS)

While both DDS Security and TLS aim to secure communications, their approaches to authentication and communication models differ significantly:

- **Mutual vs. One-Way Authentication:**
  - **DDS Security:** Implements mutual authentication, requiring both communicating parties (publishers and subscribers) to present and verify each other's certificates. This ensures that both entities are trusted and authorized.
  - **TLS (HTTPS):** Typically implements one-way authentication, where only the server presents a certificate to authenticate itself to the client. The client does not present a certificate unless client authentication is specifically configured.
- **Communication Model:**
  - **DDS Security:** Designed for multicast and publish/subscribe communication patterns, making it suitable for distributed and real-time systems like robotics.
  - **TLS (HTTPS):** Primarily designed for unicast client-server communication, which is more suitable for web browsing and similar applications.
- **Security Plugins and Flexibility:**
  - **DDS Security:** Utilizes a pluggable architecture through Security Plugins, allowing for the integration of custom encryption or signing algorithms, including post-quantum cryptographic methods.

  - **TLS (HTTPS):** Offers a fixed set of cryptographic algorithms, with limited flexibility for custom integrations.

- **Key Exchange Mechanism:**
  - **DDS Security:** Often employs Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) for key exchange, supporting both traditional and post-quantum cryptographic algorithms.
  - **TLS (HTTPS):** Primarily uses DH, ECDH, or RSA for key exchange, with limited support for post-quantum algorithms.

These differences highlight DDS Security's suitability for complex, distributed systems requiring robust, two-way authentication and flexible security configurations, which are essential in environments like robotics and IIoT.

4.4 Zenoh Security

Zenoh's security architecture is designed to ensure data integrity, confidentiality, and authentication across various network environments. At its core, Zenoh leverages Transport Layer Security (TLS) as its primary security mechanism, with additional support for the QUIC protocol.

4.4.1 Transport Layer Security (TLS) in Zenoh

Zenoh implements TLS [88] as its fundamental security protocol, offering two main configuration modes:

- **Server-side Authentication**: In this mode, clients validate the server's TLS certificate, mirroring the typical web browser-server interaction. This provides a basic level of security by ensuring the server's identity.
- **Mutual Authentication** (mTLS): This more secure mode requires both server-side and client-side authentication, ensuring that both parties in the communication are verified.

The TLS implementation in Zenoh is highly flexible, allowing for easy configuration through JSON-based files. This approach enables users to specify TLS certificates, private keys, and other security parameters. Zenoh supports both self-managed certificates and integration with established Certificate Authorities, including Let's Encrypt.

A notable feature of Zenoh's TLS implementation is its support for IP-based certificates. This functionality, introduced in recent versions, allows for generating and using certificates associated with IP addresses, not just domain names. This enhancement significantly increases deployment flexibility, especially in environments where DNS-based certificates may not be feasible.

The Zenoh Charmander 0.7.2-rc release [89] further improved TLS support by introducing Let's Encrypt support for cloud deployments and enhancing IP-based certificate handling. These updates demonstrate Zenoh's commitment to providing up-to-date security options that cater to various deployment scenarios, from local networks to cloud-based systems.

4.4.2 QUIC Protocol Support

In addition to standard TLS over TCP, Zenoh supports the QUIC protocol [90]. QUIC is a UDP-based, stream-multiplexing, encrypted transport protocol that natively embeds TLS for encryption, authentication, and confidentiality.

It's important to note that QUIC in Zenoh uses the same TLS configuration as TLS over TCP. The primary difference lies in the transport layer, not in the security features. As of the current implementation, Zenoh supports server-side authentication in QUIC, mirroring the typical web browser-server interaction. The process for generating and managing TLS certificates for QUIC is identical to that used for TLS over TCP, simplifying configuration across different transport protocols.

4.4.3 Rustls: The Core TLS Library

At the heart of Zenoh's TLS implementation is Rustls, a modern TLS library written in Rust. Rustls was chosen for its strong security properties, performance, and compatibility with Zenoh's Rust-based architecture.

Rustls provides several key advantages:

- **Memory Safety**: Being written in Rust, Rustls offers strong guarantees against common security vulnerabilities related to memory management.
- **Modern Cryptography**: Rustls is designed to avoid deprecated or insecure cryptographic algorithms and protocols, ensuring that only up-to-date and secure methods are used.
- **Performance Optimization**: Rustls is highly optimized, aligning well with Zenoh's focus on efficient data distribution.
- **Flexible Cryptographic Backends**: Rustls supports various cryptographic backends, allowing Zenoh to adapt to different regulatory or performance requirements.

The integration of Rustls ensures that all TLS connections in Zenoh, whether over TCP or as part of QUIC, benefit from a modern, secure, and efficient implementation. This is important for maintaining the security and integrity of data as it flows through Zenoh networks, especially in distributed and potentially untrusted environments.

In summary, Zenoh's security framework, built on the foundation of Rustls and enhanced with features like Let's Encrypt support and IP-based certificates, provides a flexible solution for securing data communication in diverse network environments. This approach ensures that Zenoh can meet the security requirements of a wide range of applications, from local development to large-scale, distributed systems in cloud environments.

4.4.4 Post-Quantum Cryptography

As Zenoh relies on Rustls for TLS transport, the integration of Post-Quantum Cryptography (PQC) in Zenoh is closely tied to PQC support in Rustls. Recent developments in this area present both opportunities and challenges for enhancing Zenoh's security against potential quantum threats.

PQC Support in Rustls The `rustls-post-quantum` crate introduces experimental Post-Quantum Cryptography (PQC) support within Rustls [91]. This crate implements a hybrid key exchange mechanism, combining classical and post-quantum algorithms to defend against both current and future quantum adversaries [92]. At its core is the X25519Kyber768Draft00 algorithm [93], which merges the well-established classical X25519 algorithm with Kyber768Draft00, a post-quantum key encapsulation mechanism (KEM). This hybrid approach aims to maintain security even if one component is compromised, providing protection against both classical and quantum attacks. However, as a pre-standardization implementation, users should be aware of potential interoperability issues. The crate can be incorporated into Rustls as either a default or custom crypto provider, offering flexibility in integration. To utilize this PQC support in a Rust-based application, developers need to configure Rustls to use the rustls-post-quantum provider at the application level. This involves creating a custom CryptoProvider with the hybrid key exchange and installing it as the default provider. While this process offers enhanced security, it requires careful implementation and consideration of the experimental nature of the technology.

```
use rustls::crypto::{aws_lc_rs, CryptoProvider};
use rustls_post_quantum::X25519Kyber768Draft00;

// Create a custom CryptoProvider with hybrid key exchange
let parent = aws_lc_rs::default_provider();
let my_provider = CryptoProvider {
    kx_groups: vec![
        &X25519Kyber768Draft00,
        aws_lc_rs::kx_group::X25519,
    ],
    ..parent
};

// Install the custom provider as the default
rustls_post_quantum::provider().install_default().unwrap();
```

Implications for Zenoh The development of PQC support in Rustls presents promising opportunities for the integration of PQC into Zenoh's security. However, the practical integration of the rustls-post-quantum provider requires explicit enabling at the application level, necessitating modifications to Zenoh's codebase.

Looking ahead, while the potential for improved security is significant, the path to incorporating PQC into Zenoh remains a subject for future research and development.

4.5 SROS2: Secure Robot Operating System

SROS2 (Secure Robot Operating System 2) [94, 34, 95] is a suite of security tools and libraries designed to bolster the security of ROS 2 systems. Recognizing the increasing importance of cybersecurity in robotics, SROS2 aims to provide robust security measures that are seamlessly integrated into the ROS 2 development process.

4.5.1 SROS2 Architecture and Features

SROS2 leverages the security capabilities of the underlying Data Distribution Service (DDS) middleware used in ROS 2. By building upon DDS-Security, SROS2 introduces authentication and access control mechanisms for ROS 2 nodes and topics.

A key component of SROS2 is its integration with the ROS Client Library (RCL), which manages the security files required by DDS-Security for each domain participant. These security artifacts include certificates and keys necessary for authenticating participants and encrypting communications. SROS2 simplifies the generation and management of these artifacts through dedicated tools.

The concept of enclaves is central to SROS2's approach. Enclaves are logical groupings of nodes that share a common security context. Developers can assign nodes to specific enclaves, and SROS2 will handle the configuration and distribution of the necessary security files. This organization aids in managing complex systems by structuring security settings hierarchically.

To implement fine-grained security policies, SROS2 provides tools for generating governance and permissions files:

- Governance File: Defines the security policies for the DDS domain, including encryption requirements and access control settings.
- Permissions File: Specifies the access rights of each domain participant, controlling which nodes can publish or subscribe to particular topics or services.

SROS2 supports two security strategies:

- Permissive Mode: If security artifacts are missing for a node, it runs without security features enabled.
- Enforce Mode: The node fails to run if the required security files are absent, ensuring strict security compliance.

The SROS2 command-line interface (CLI) enhances usability by providing commands like ros2 security. This tool assists users in creating keystores, generating keys and certificates, and managing security artifacts, abstracting the complexities of Public Key Infrastructure (PKI) and DDS-Security configurations.

By integrating these security features through environment variables and command-line arguments, SROS2 allows developers to enhance security without significant alterations to their existing codebases.

4.5.2 Current Limitations and Challenges

Despite its advancements, SROS2 encounters several limitations:

- Interoperability Issues: Secure communication between different DDS vendor implementations is currently unsupported. Since DDS-Security implementations can be vendor-specific, this limitation affects heterogeneous systems where nodes might use different DDS vendors.

- Complexity in Security Management: Setting up and managing security artifacts requires an understanding of PKI and DDS-Security. While SROS2 provides tools to ease this process, the learning curve remains steep for developers unfamiliar with cybersecurity principles.

- Artifact Management Overhead: In large or distributed systems, keeping security artifacts like certificates and keys updated across all nodes is challenging. Any lapses can compromise the system's security and functionality.

- Static Graph Modeling: SROS2's reliance on static snapshots of the computation graph may overlook dynamic resource access events, such as transient service clients or action requests. This oversight can lead to incomplete security policies, leaving vulnerabilities unaddressed.

- Enclave Configuration Complexity: Deciding how to partition nodes into enclaves requires careful consideration of security requirements and trust boundaries. The lack of automated tools or clear guidelines makes this process complex and potentially error-prone.

- Middleware Compatibility: SROS2 is primarily designed for the default DDS middleware. Limited compatibility with other communication middlewares can hinder integration with non-ROS systems or specialized applications requiring different middleware solutions.

Addressing these challenges is important for the evolution of SROS2. Efforts are underway to improve interoperability between DDS implementations, enhance the user-friendliness of security tools, and support more dynamic system architectures. Overcoming these hurdles will significantly strengthen the security of ROS 2-based robotic systems, particularly as they are deployed in increasingly sensitive and mission-critical environments.

5 Enhancing ROS 2 Security with Post-Quantum Cryptography

This section presents an analysis of the implementation of Post-Quantum Cryptography (PQC) integration into ROS 2. This work addresses the need for quantum-resistant security measures in robotic systems, focusing on two main approaches to enhance ROS 2's cryptographic capabilities.

First, we analyze the integration of PQC using our custom plugin, PQSec-DDS, developed for Cyclone DDS, alongside a modified version of SROS2. This approach focuses on adapting the core cryptographic functionalities of DDS to support post-quantum algorithms while maintaining compatibility with current ROS 2 security management.

Secondly, we explore the integration of PQC into ROS 2 systems using Zenoh. We analyze two scenarios:

- As a bridge for wide-area network (WAN) communications with the Zenoh bridge.
- As a native middleware for ROS 2 with the new `rmw_zenoh`, providing an alternative approach to implementing PQC directly within the entire communication layer.

This work is supported by the following open-source repositories, which provide the necessary implementations and modifications for integrating Post-Quantum Cryptography into ROS 2:

- **PQSec-DDS** [87]: A custom plugin developed for Cyclone DDS that integrates post-quantum cryptography algorithms into the DDS Security specification.
- **PQC-enhanced SROS2** [96]: A fork of the original SROS2 implementation, extended to support post-quantum cryptography using the Open Quantum Safe (OQS) provider for OpenSSL.
- **PQC-enabled Zenoh** [97]: A fork of the Zenoh middleware, modified to incorporate post-quantum cryptography algorithms through the rustls-post-quantum provider for Rustls, enabling quantum-resistant TLS connections.

5.1 Extending SROS2 for PQC Support

Our extension of the original SROS2 implementation [98] is publicly available as an open-source contribution in a fork [96]. This fork demonstrates a practical integration of PQC into the existing SROS2 infrastructure, providing a reference implementation for potential future incorporation into the main SROS2 codebase and facilitating community review and further development.

The original SROS2 module is written in Python and utilizes the cryptography library [99] for its cryptographic operations. This library provides a set of cryptographic primitives and high-level functions for Python developers. However, it currently does not support PQC. Hence, we made direct OpenSSL calls, specifically using the Open Quantum Safe (OQS) provider for OpenSSL [78] that integrate the cryptographic library liboqs into OpenSSL [77]. This approach allowed us to leverage post-quantum algorithms that are not yet available in standard Python cryptographic libraries. The integration involved creating wrapper classes and functions to interface with OpenSSL's PQC implementations, ensuring that the existing SROS2 architecture could seamlessly incorporate these new quantum-resistant algorithms.

The first step in integrating PQC support was to extend the SROS2 command-line interface. We introduced a new parameter, `-pq-algorithm`, to allow users to specify a post-quantum algorithm when creating keystores or enclaves. This modification enables seamless integration of PQC options into existing SROS2 workflows. For example:

```
ros2 security create_keystore /path/to/keystore --pq-algorithm dilithium3
ros2 security create_enclave /path/to/keystore my_enclave \
                            --pq-algorithm dilithium3
```

The primary modifications were made to the core cryptographic module, `utilities.py`. The main high level changes involved:

- **Key and Certificate Generation**: The `build_key_and_cert` function was updated to support post-quantum algorithms when specified and available. It now includes logic to generate PQ keys and certificates using direct OpenSSL calls when appropriate.
- **S/MIME Signing**: The `create_smime_signed_file` function was enhanced to support post-quantum signatures, ensuring that S/MIME signing operations leverage post-quantum cryptographic methods when enabled. This implementation follows the approach outlined by the Open Quantum Safe project for CMS and S/MIME operations using post-quantum algorithms [100]. The integration utilizes the OQS provider with OpenSSL 3, allowing for the creation and verification of quantum-safe digital signatures in the Cryptographic Message Syntax (CMS) format, which is the foundation of S/MIME.
- **Provider Integration**: Integration with the Open Quantum Safe (OQS) provider was established to facilitate post-quantum cryptographic operations. This involved configuring OpenSSL to use the OQS provider and ensuring that cryptographic calls are routed through this provider.
- **Fallback Mechanism**: A fallback mechanism was implemented to revert to traditional cryptographic methods if the post-quantum provider is unavailable, ensuring compatibility.

An example of the modified logic in `utilities.py`:

```
def build_key_and_cert(subject_name,
    *,
    ca=False,
    ca_key=None,
    issuer_name='',
    pq_algorithm='default'
    ):
    use_pq = pq_algorithm != 'default' and is_provider_available(OQS_PROVIDER_NAME)

    if use_pq:
        # Use post-quantum cryptographic algorithms
        private_key = generate_pq_key(pq_algorithm=pq_algorithm)
        # ... PQ certificate generation logic
    else:
        # Fallback to traditional cryptographic algorithms
        private_key = ec.generate_private_key(ec.SECP256R1(),
                                              cryptography_backend()
                                              )
        # ... Traditional certificate generation logic

    return (cert, private_key)
```

The centralized handling of cryptographic operations in `utilities.py` allowed for minimal changes in other dependent modules, while still extending PQC support throughout the SROS2 system. Key modules affected include:

- `enclave.py`: Functions for enclave creation and management, such as `create_enclave` and `_create_key_and_cert`, now seamlessly incorporate PQC capabilities through their use of the updated utility functions.
- `_keystore.py`: Keystore operations, including CA certificate generation and governance file signing, leverage PQC when enabled without requiring direct modifications to the module itself.
- `_permission.py`: The creation and signing of permission files now benefit from PQC support, ensuring quantum-resistant security for these components when the feature is activated.

This approach ensures a consistent and user-friendly integration of PQC options across the entire SROS2 ecosystem. From programmatic APIs to command-line tools and build system integrations, the implementation maintains a uniform interface while providing enhanced security options. This design simplifies the adoption of PQC within existing ROS 2 workflows and also facilitates future expansions and modifications to the security framework.

5.1.1 Compatibility with PQSec-DDS

The PQC-enhanced SROS2 framework is designed to work in conjunction with PQC-enabled DDS implementations, specifically PQSec-DDS [87], a post-quantum cryptography plugin for the CycloneDDS middleware. This compatibility allows for a comprehensive post-quantum security solution in ROS 2 systems.

Complementary Roles SROS2 and PQSec-DDS serve complementary roles in providing post-quantum security:

- **SROS2**: Manages security artifacts (certificates, keys, etc.) and supports post-quantum signatures for these artifacts.
- **PQSec-DDS**: Handles secure communication at the DDS level, implementing both post-quantum Key Encapsulation Mechanisms (KEMs) and signatures for actual data exchange.

This combination ensures that both the security infrastructure (managed by SROS2) and the data communication channels (secured by PQSec-DDS) are resistant to quantum attacks.

5.1.2 Testing and Validation

To fully validate the post-quantum security enhancements, it is necessary to test the integration of PQC-enhanced SROS2 with CycloneDDS and the PQSec-DDS plugin. This testing ensures that the entire stack, from security artifact generation to secure communication, is utilizing post-quantum cryptography. A detailed documentation can be found in the `sros2_pqsecdds.md` file of our repository [96].

Testing Setup The testing environment requires the following components:

- PQC-enhanced SROS2 fork
- CycloneDDS middleware
- PQSec-DDS plugin for CycloneDDS
- Necessary dependencies: liboqs, oqs-provider

Testing Procedure The high-level steps for testing include:

1. Setting up the environment with all necessary dependencies.
2. Building and configuring the PQSec-DDS plugin for CycloneDDS.
3. Building the PQC-enhanced SROS2 fork.
4. Generating security artifacts using SROS2 with PQ algorithms.
5. Configuring CycloneDDS to use the PQSec-DDS plugin.
6. Running test nodes (talker and listener) with PQ-secured communication.

Key Test Scenarios A specific test scenario was developed to ensure the correct functioning of the integrated system:

- Generating a keystore with post-quantum algorithms:

```
ros2 security create_keystore demo_keystore --pq-algorithm dilithium3
```

- Creating enclaves with post-quantum security:

```
ros2 security create_enclave demo_keystore /talker_listener/talker \
    --pq-algorithm dilithium3

ros2 security create_enclave demo_keystore /talker_listener/listener \
    --pq-algorithm dilithium3
```

- Secure communication between nodes using PQC:

```
ros2 run demo_nodes_cpp talker \
--ros-args --enclave /talker_listener/talker
ros2 run demo_nodes_py listener \
--ros-args --enclave /talker_listener/listener
```

This validation process verifies the generation of security artifacts using PQ algorithms, correct utilization of the PQSec-DDS plugin, secure node communication with PQC, and fallback to traditional cryptography when necessary.

5.2 Secure Communication with the Zenoh Bridge

Having explored the integration of PQC into ROS 2 through our custom PQSec-DDS plugin and modified SROS2, we now turn our attention to the second approach: leveraging Zenoh for secure, quantum-resistant communication. This section focuses on utilizing Zenoh as a bridge for wide-area network (WAN) communication in ROS 2 systems, examining how post-quantum cryptography can be integrated into this setup.

In this context, two primary approaches for securing communication emerge, each with its own set of advantages and considerations, particularly in relation to DDS security and post-quantum readiness. A relevant aspect of these approaches is the authentication model employed, which builds upon the DDS authentication process discussed in Section 4.3.2.

5.2.1 Approach 1: Transporting Encrypted DDS Messages over Zenoh

This approach involves encrypting ROS 2 messages at the DDS level using SROS2 before they are transmitted through the Zenoh bridge. Key characteristics include:

- **End-to-End Encryption**: Messages are encrypted by the ROS 2 nodes before reaching the Zenoh bridge and remain encrypted until they reach their destination nodes.
- **Certificate Distribution**: Requires distribution of SROS2 certificates to all participating ROS 2 nodes, including those on separate machines connected via Zenoh.
- **Zenoh Transport Options**: The Zenoh bridge can use TCP, TLS, or QUIC for transport, adding an optional layer of transport security.
- **Post-Quantum Considerations**: With our PQC-enhanced SROS2, this method can utilize post-quantum algorithms for the DDS-level encryption, making the entire communication chain quantum-resistant. Additionally, the TLS-enabled Zenoh transport can be secured with post-quantum providers.
- **Mutual Authentication Impact**: Leveraging DDS's mutual authentication ensures that both the publisher and subscriber verify each other's identities across the internet. This bidirectional verification enhances security but requires robust certificate management and synchronization across all remote nodes.

This method maintains the security model of SROS2 across the Zenoh bridge, ensuring that messages remain confidential and integrity-protected throughout their transport. However, it necessitates careful management of certificates across all nodes in the system, which can be operationally intensive in large-scale or dynamic environments.

5.2.2 Approach 2: Transporting Unencrypted DDS Messages over Encrypted Zenoh

In this approach, ROS 2 messages are transmitted unencrypted at the DDS level but are secured by encrypting the Zenoh communication layer. Key aspects include:

- **Transport-Level Encryption**: Security is provided by the Zenoh layer, typically using TLS or QUIC protocols.
- **Simplified ROS 2 Configuration**: Does not require SROS2 setup for ROS 2 nodes, simplifying the configuration of individual nodes.
- **Centralized Security Management**: Security is managed at the Zenoh bridge level, potentially simplifying certificate management for large-scale systems.
- **Post-Quantum Adaptability**: Requires integration of post-quantum algorithms at the Zenoh level to achieve quantum resistance.
- **Authentication Options**: Zenoh supports both one-way and mutual authentication (mTLS) [88]. While one-way authentication is common, where only the Zenoh bridge authenticates to the ROS 2 nodes, mutual authentication can be implemented for enhanced security.

It's important to note that Zenoh's support for mutual authentication (mTLS) provides an option for bidirectional verification similar to the DDS approach. This feature requires proper configuration of certificates for both the client and server sides of the Zenoh communication [88]. The implementation of mTLS in Zenoh can significantly enhance the security posture, addressing some of the concerns associated with one-way authentication.

This method offloads the encryption responsibility to the Zenoh layer, which can be advantageous in scenarios where configuring individual ROS 2 nodes with SROS2 is impractical or when a unified security approach across different types of data (not just ROS 2 messages) is desired.

5.2.3 Comparison and Operational Implications

The selection between transporting encrypted DDS messages over Zenoh and transporting unencrypted DDS messages over encrypted Zenoh hinges on the specific security requirements, scalability needs, and operational constraints of the system, as summarized in Table 1.

- **Encrypted DDS over Zenoh** aligns closely with the post-quantum enhancements implemented in SROS2, providing robust end-to-end security through mutual authentication. This approach is ideal for scenarios demanding high security and trust between all communicating parties, despite the increased complexity in certificate management.
- **Unencrypted DDS over Encrypted Zenoh** offers a streamlined configuration and centralized security management, making it suitable for large-scale or heterogeneous systems where managing individual node security is impractical. However, it necessitates additional efforts to incorporate post-quantum algorithms at the transport layer to ensure quantum resistance.

In the context of our post-quantum security enhancements, the first approach allows immedi-

Encrypted DDS over Zenoh	**Unencrypted DDS over Encrypted Zenoh**
Encryption Scope End-to-end security at message level	**Encryption Scope** Security at transport level
Authentication Model **Mutual Authentication**: Both publisher and subscriber authenticate each other via SROS2 certificates	**Authentication Model** **Flexible Authentication**: Supports both one-way and mutual authentication (mTLS) at the Zenoh level
Certificate Management Requires distribution and management of SROS2 certificates across all ROS 2 nodes	**Certificate Management** Centralized certificate management at Zenoh bridges; can be extended to nodes if using mTLS
Transport Flexibility Supports multiple Zenoh transport options (TCP/TLS/QUIC), allowing layered security	**Transport Flexibility** Utilizes Zenoh's transport-level security (TLS/QUIC)
Post-Quantum Integration Integrated at DDS level with PQC-enhanced SROS2 and optional PQC in Zenoh transport	**Post-Quantum Integration** Requires separate PQC integration in Zenoh transport layer
Operational Complexity Higher due to certificate management and synchronization across all ROS 2 nodes	**Operational Complexity** Variable: Lower with one-way auth, moderate with mTLS

Table 1: Comparison of Secure Communication Approaches with Zenoh

ate utilization of our PQC-enabled SROS2, ensuring that all DDS communications are inherently quantum-resistant. The second approach, while simplifying node configurations, would require extending post-quantum integrations to the Zenoh transport layer, presenting a promising direction for future research and development.

5.3 Secure Communication with the Zenoh Middleware

Beyond bridging ROS 2 communication over Zenoh, an alternative approach involves using Zenoh as a native middleware for ROS 2 nodes. This section explores how secure communication is achieved when ROS 2 nodes communicate directly over Zenoh, both in local and remote environments. The integration is in our Zenoh fork [97]. Further documentation can be found in the `test_pq_ros2.md`

Utilizing Zenoh as the middleware layer in ROS 2 replaces the default DDS middleware, allowing nodes to communicate using Zenoh protocols directly. This integration offers several benefits, including the elimination of the DDS-Zenoh bridging requirement, which reduces latency and complexity. Zenoh's support for various transport protocols such as TCP, UDP, and QUIC provides flexibility, optimizing performance across different network environments. Moreover, Zenoh efficiently handles communication over local and wide-area networks, enhancing scalability for distributed systems.

When Zenoh is employed as the middleware, the security mechanisms provided by DDS and SROS2 are no longer directly applicable. Instead, security must be managed within the Zenoh framework. Zenoh supports Transport Layer Security (TLS) for encrypting communications between nodes and offers mutual TLS authentication (mTLS), ensuring that both clients and servers authenticate each other. This bidirectional verification enhances trust and security across the network. Additionally, Zenoh's use of Rustls allows for the integration of post-quantum cryptography (PQC) algorithms, providing quantum-resistant security at the transport layer.

5.3.1 Modifications to Zenoh for PQC Support

To enable Post-Quantum Cryptography (PQC) within Zenoh, specific modifications are required at the application level. Our work, documented in our Zenoh fork [97], focuses on integrating PQC support by leveraging recent developments in Rustls, the TLS library used by Zenoh for secure communications.

For this We incorporate the `rustls-post-quantum` crate [91], which provides experimental PQC support within Rustls. This has to be done at the application level, we create and install a custom CryptoProvider that includes the hybrid key exchange algorithm X25519Kyber768Draft00 [93].

The details of our implementation and testing procedures are documented in the `test_pq_ros2.md` file within our repository [97].

5.3.2 Leveraging SROS2 for Certificate Generation and Mutual Authentication

Although SROS2 is designed around DDS security and does not natively support non-DDS middlewares like Zenoh, we can harness SROS2's capabilities for certificate generation to facilitate mutual authentication within Zenoh's both TLS and mTLS setup. Here we provide the instruction for testing the later. By using SROS2 tools to generate the necessary X.509 certificates, we maintain a consistent certificate management process across the system.

The steps involve:

- **Using SROS2 for Certificate Generation**: Utilize SROS2's `generate_artifacts` command to create certificates for each Zenoh participant, including nodes and routers. This provides a familiar and standardized method for certificate generation.

- **Configuring Zenoh with Generated Certificates**: Configure Zenoh's nodes and routers to use the certificates generated by SROS2 for mutual TLS authentication. This involves specifying the paths to the certificates and keys in Zenoh's configuration files.

- **Ensuring Certificate Compatibility**: Since both Zenoh and SROS2 use standard X.509 certificates, the certificates generated by SROS2 are compatible with Zenoh's TLS implementation.

By integrating SROS2's certificate management with Zenoh's mTLS capabilities, we streamline the security setup process and enable mutual authentication between Zenoh participants.

5.3.3 Implementing Secure Communication over Zenoh

To secure ROS 2 communication over Zenoh with PQC and mTLS, follow these steps:

- **Install Zenoh and RMW Packages**:
  - Install the Zenoh router and client libraries from our fork [97]

  - Install the `rmw_zenoh_cpp` package for ROS 2 integration

- **Generate PQC-Enabled TLS Certificates**:

  - Use SROS2's certificate generation tools:

```
ros2 security create_keystore /keystore \
--pq-algorithm dilithium3
ros2 security create_enclave /keystore zenoh_router \
--pq-algorithm dilithium3
ros2 security create_enclave /keystore zenoh_client \
--pq-algorithm dilithium3
```

  - Certificates will be located in `/keystore/enclaves/`

- **Configure Zenoh Router with mTLS**: Edit the Zenoh router configuration file:

```
{
  "mode": "router",
  "listen": {
    "endpoints": ["tls/localhost:7447"]
  },
  "transport": {
    "link": {
      "tls": {
        "root_ca_certificate": \
            "/keystore/enclaves/zenoh_client/cert.pem",
        "client_auth": true,
        "server_private_key": \
            "/keystore/enclaves/zenoh_router/private/key.pem",
        "server_certificate": \
            "/keystore/enclaves/zenoh_router/cert.pem"
      }
    }
  }
}
```

- **Configure Zenoh Clients**:

  - Set up client configuration:

```
{
  "mode": "client",
  "connect": {
    "endpoints": ["tls/localhost:7447"]
  },
  "transport": {
    "link": {
      "tls": {
        "root_ca_certificate": \
            "/keystore/enclaves/zenoh_router/cert.pem",
        "client_auth": true,
        "client_private_key": \
            "/keystore/enclaves/zenoh_client/private/key.pem",
        "client_certificate": \
            "/keystore/enclaves/zenoh_client/cert.pem"
      }
    }
  }
}
```

- **Set Up ROS 2 Nodes with Zenoh**:
  - Configure ROS 2 nodes to use Zenoh:

```
export RMW_IMPLEMENTATION=rmw_zenoh_cpp
export ZENOH_CONFIG=/path/to/zenoh_client_config.json
```

- **Launch Nodes and Router**:
  - Start the Zenoh router with PQC and mTLS enabled
  - Launch ROS 2 nodes, ensuring they use the Zenoh middleware and establish secure, quantum-resistant connections

5.3.4 Testing and Validation

Testing and validation involve verifying that nodes can establish PQC-secured connections both locally and remotely. Perform local testing to ensure secure communication between nodes on the same machine or local area network (LAN). Conduct remote testing over wide area networks (WAN) to confirm that PQC-enabled TLS encryption and mutual authentication are maintained across different network environments. Utilize Zenoh's built-in logging and monitoring tools or network analyzers to confirm that data transmission is encrypted using PQC algorithms.

5.3.5 Advantages and Considerations

Using Zenoh as the native middleware for ROS 2 with PQC and mTLS offers several advantages. It provides quantum-resistant security at the transport layer, enhancing the overall security of the system against future quantum threats. Leveraging SROS2 for certificate generation streamlines the security setup process, allowing us to reuse existing tools and practices. Mutual authentication ensures that both clients and servers are authenticated, enhancing trust and security across the network.

However, there are considerations to be aware of:

- **PQC Maturity**: The integration of PQC algorithms into Rustls and Zenoh is still evolving, potentially affecting stability and support.

- **Operational Overhead**: Managing and distributing PQC-enabled certificates introduces additional complexity, especially in large deployments.
- **SROS2 Limitations**: While SROS2 assists with certificate generation, its other security features are not directly applicable to Zenoh, requiring adjustments in security management strategies.

6 Conclusions and Future Work

This thesis has explored the integration of Post-Quantum Cryptography (PQC) into ROS2, addressing the need for quantum-resistant security in robotic systems. Our work has focused on two main approaches: enhancing SROS2 with PQC capabilities and adapting Zenoh for quantum-resistant communication. Through these efforts, we have made several contributions to the field of robotics security:

- Extended SROS2 to support PQC by integrating the Open Quantum Safe (OQS) provider for OpenSSL, enabling the generation of quantum-resistant keys and certificates.
- Modified Zenoh to incorporate PQC algorithms, leveraging Rustls to establish quantum-resistant TLS connections.
- Demonstrated compatibility between our PQC-enhanced SROS2 and our previously developed PQSec-DDS.
- Established secure, quantum-resistant communication channels between ROS 2 nodes using Zenoh both as a native middleware with mTLS and as a bridge with DDS security.

These contributions represent a significant step towards preparing ROS2 and its associated middleware for the post-quantum transition.

Throughout this thesis we gained important insights that shed light on the challenges and opportunities in implementing PQC in robotics systems. We found that the integration of PQC into DDS closely mirrors that of TLS/DTLS, allowing lessons from these protocols to be applied in the robotics domain. The DDS Security Specification SPI's API design proved advantageous, facilitating the creation of external plugins and enabling the crypto-agility necessary for PQC adoption. However, we also noted that not all open-source DDS vendors currently provide straightforward external plugin integration, which could potentially complicate PQC adoption in some implementations.

Our work with Zenoh revealed several important insights. One of the key advantages of Zenoh is its utilization of popular security mechanisms based on the TLS and QUIC transports. This approach eliminates the need for an additional security specification, unlike DDS which requires its own security standard. Consequently, Zenoh can potentially benefit more readily from advancements in these widely-used protocols. However, Zenoh's reliance on Rustls, rather than OpenSSL, presents a current limitation. While OpenSSL has made significant progress in PQC integration through projects like Open Quantum Safe, the PQC support in Rustls is still in a more experimental stage. This dependence on Rustls currently constrains full PQC support in Zenoh, particularly for quantum-resistant signatures. More broadly, we observed that the experimental nature of PQC integration in fundamental cryptographic libraries currently bottlenecks the transition of dependent systems to quantum-resistant security, with Rustls-dependent systems like Zenoh facing additional challenges in this regard.

Our work enhances the security infrastructure of ROS2 against potential quantum attacks, providing a foundation for quantum-resistant robotics applications. Despite these advancements, several challenges and areas for future work remain:

- Develop tools for unified security management across different middleware platforms

(DDS and Zenoh), including contributions to SROS2 to accommodate this unified approach.

- Further enhance PQC support within middleware platforms, particularly Zenoh.
- Conduct comprehensive benchmarking of PQC integration in Cyclone DDS within ROS2, focusing on scalability and performance comparisons with alternative ROS2 middlewares and traditional cryptography.

In conclusion, this thesis demonstrates significant progress in preparing ROS2 and DDS-based robotic systems for the post-quantum era. Our work on integrating PQC into SROS2, Zenoh, and DDS lays a strong foundation for quantum-resistant security in robotics. While challenges remain, particularly in terms of standardization, performance optimization, and full integration across all components, the path forward is clear.

As quantum computing continues to advance, the importance of quantum-resistant security in robotics cannot be overstated. Our research provides a step forward towards ensuring that robotic systems remain secure in the face of emerging quantum threats, paving the way for the long-term viability and trustworthiness of robotic applications across various domains.

References

[1] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[2] Open Robotics. ROS - Robot Operating System. https://www.ros.org, 2024. ROS is a set of software libraries and tools for building robot applications, offering a wide range of drivers, algorithms, and developer tools. All components are open source. Accessed: September 2024.

[3] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics*, 7(66):eabm6074, 2022.

[4] Hodei Olaizola. Alias Robotics asegura una prestigiosa financiación del Consejo Europeo de Innovación. https://news.aliasrobotics.com/asegura-unada-prestigiosa-financiacion-del-eic/, March 2024. Alias Robotics, Robot cybersecurity news. Accessed: March 2024.

[5] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[6] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[7] M. Mosca and M. Piani. Global Risk Institute: Quantum Threat Timeline Report 2022. https://globalriskinstitute.org/publication/2022-quantum-threat-timeline-report/, 2022. Accessed: 6 march 2024.

[8] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[9] National Cyber Security Centre (NCSC). Quantum security technologies. https://www.ncsc.gov.uk/pdfs/whitepaper/quantum-security-technologies.pdf, Mar 2020. Accessed: 6 march 2024.

[10] National Security Agency (NSA). Quantum key distribution (qkd) and quantum cryptography (qc). https://www.nsa.gov/Cybersecurity/Quantum-Key-Distribution-QKD-and-Quantum-Cryptography-QC/. Accessed: 2024-02-15.

[11] NSA CISA and NIST. Quantum-readiness: Migration to post-quantum cryptography. https://media.defense.gov/2023/Aug/21/2003284212/-1/-1/0/CSI-QUANTUM-READINESS.PDF, Aug 2023. Accessed: 15 february 2024.

[12] Centro Criptológico Nacional (CCN)-PYTEC. Recomendaciones para una transición postcuántica segura, December 2022. CCN-TEC 009.

[13] French Cybersecurity Agency (ANSSI), Federal Office for Information Security (BSI),

Netherlands National Communications Security Agency (NLNCSA), and Swedish Armed Forces Swedish National Communications Security Authority. Position paper on quantum key distribution. Technical report, January 2024.

[14] National Cyber and Information Security Agency (NÚKIB). Letter of Support to the Position Paper on Quantum Key Distribution. https://nukib.gov.cz/en/infoservis-en/news/2163-letter-of-support-to-the-position-paper-on-quantum-key-distribution/, September 2024. NÚKIB expresses support for the transition to post-quantum cryptography and the continued research on quantum key distribution. Accessed: September 2024.

[15] National Institute of Standards and Technology (NIST). Announcing Issuance of Federal Information Processing Standards (FIPS) FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard, FIPS 204, Module-Lattice-Based Digital Signature Standard, and FIPS 205, Stateless Hash-Based Digital Signature Standard, August 2024.

[16] Deployment of Post Quantum Cryptography in systems in industrial sectors. https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/digital-eccc-2024-deploy-cyber-06-pqcindustry, January 2024. Call for proposals under the Digital Europe Programme (DIGITAL). Call ID: DIGITAL-ECCC-2024-DEPLOY-CYBER-06-PQCINDUSTRY. Deadline: March 26, 2024. Accessed: September 2024.

[17] Post-quantum cryptography transition. https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/horizon-cl3-2024-cs-01-02, June 2024. Call for proposals under the Horizon Europe Programme (HORIZON). Call ID: HORIZON-CL3-2024-CS-01-02. Deadline: November 20, 2024. Accessed: September 2024.

[18] M Diaz-Cacho, A Barreiro, and Matías García Rivera. Bidireccionalidad y eficiencia en el transporte de datos de teleoperación a través de redes ip. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 7(2):99–110, 2010.

[19] Ronald L Krutz. *Securing SCADA systems*. John Wiley & Sons, 2015.

[20] KS Manoj. *Industrial automation with SCADA: concepts, communications and security*. Notion Press, 2019.

[21] Günter Niemeyer, Carsten Preusche, Stefano Stramigioli, and Dongjun Lee. Telerobotics. *Springer handbook of robotics*, pages 1085–1108, 2016.

[22] Dezhen Song, Ken Goldberg, and Nak-Young Chong. Networked robots. In *Springer Handbook of Robotics*, pages 1109–1134. Springer, 2016.

[23] JM Azorín Poveda. Control bilarteral por convergencia de estados de sistemas teleoperados con retardos en las transmisión. *Universidad Miguel Hernandez, Elche*, 2003.

[24] Beatriz Fernández Muro. Securing communications in surgery robots. 2018.

[25] George Kokkonis, Kostas E Psannis, Sotirios Kontogiannis, Petros Nicopolitidis, Manos Roumeliotis, and Yutaka Ishibashi. Interconnecting haptic interfaces with high update rates through the internet. *Applied System Innovation*, 1(4):51, 2018.

[26] Raul Wirz, Raul Marín, José M Claver, Manuel Ferre, Rafael Aracil, and Josep Fernández. End-to-end congestion control protocols for remote programming of robots, using heterogeneous networks: A comparative analysis. *Robotics and Autonomous Systems*, 56(10):865–874, 2008.

[27] Jean-Philippe Vasseur and Adam Dunkels. *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

[28] George Bekey and Junku Yuh. The status of robotics. *IEEE Robotics & Automation Magazine*, 15(1):80–86, 2008.

[29] Ivan Mezei, Veljko Malbasa, and Ivan Stojmenovic. Robot to robot. *IEEE robotics & automation magazine*, 17(4):63–69, 2010.

[30] Lynne E Parker, Daniela Rus, and Gaurav S Sukhatme. Multiple mobile robot systems. *Springer handbook of robotics*, pages 1335–1384, 2016.

[31] J Ernesto Solanes, Adolfo Munoz, Luis Gracia, Ana Martí, Vicent Girbés-Juan, and Josep Tornero. Teleoperation of industrial robot manipulators based on augmented reality. *The International Journal of Advanced Manufacturing Technology*, 111:1077–1097, 2020.

[32] Alessandro Manzi, Laura Fiorini, Raffaele Limosani, Peter Sincak, Paolo Dario, and Filippo Cavallo. Use case evaluation of a cloud robotics teleoperation system (short paper). In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, pages 208–211. IEEE, 2016.

[33] Raúl Lozano Teruel. Real-time haptic communications for immersive telerobotics. 2022.

[34] Victor Mayoral-Vilches, Ruffin White, Gianluca Caiazza, and Mikael Arguedas. Sros2: Usable cyber security tools for ros 2. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11253–11259. IEEE, 2022.

[35] William Woodall. ROS on DDS. https://design.ros2.org/articles/ros_on_dds.html, Jun 2014. ROS 2 Design Article. GitHub: ros2/design, commit 12f61b14698b80170824c699c70608d9ded3a6d7. Available at https://github.com/ros2/design/blob/12f61b14698b80170824c699c70608d9ded3a6d7/articles/020_ros_with_dds.md.

[36] ROS Wiki. ROS Technical Overview. https://wiki.ros.org/ROS/Technical%20Overview, 2014. Accessed on 2024-03-25.

[37] ROS Wiki. TCPROS. http://wiki.ros.org/ROS/TCPROS, 2013. Accessed on 2024-03-25.

[38] Data Distribution Service Version 1.4. https://www.omg.org/spec/DDS/1.4/, March 2015. Describes the Data-Centric Publish-Subscribe (DCPS) model for efficient and robust data delivery.

[39] Gerardo Pardo-Castellote. OMG Data-Distribution Service: Architectural Overview. Real-Time Innovations, Inc., 2024. Contact email: gerardo@rti.com.

[40] DDS Interoperability Wire Protocol Version 2.5. https://www.omg.org/spec/DDSI-RTPS/2.5/, April 2022. Describes the Real-Time Publish Subscribe Protocol for DDS interoperability.

[41] OpenDDS Developers. Introduction to OpenDDS. https://opendds.readthedocs.io/en/dds-3.29.1/devguide/introduction.html#introduction-what-is-opendds, 2024. OpenDDS Documentation. GitHub: OpenDDS/OpenDDS, commit d80b91ea64ed0458fb173db21277a8a2f9ec7134. Available at https://github.com/OpenDDS/OpenDDS/blob/d80b91ea64ed0458fb173db21277a8a2f9ec7134/docs/devguide/introduction.rst.

[42] ROS 2 Documentation. Different ROS 2 Middleware Vendors. https://docs.ros.org/en/jazzy/Concepts/Intermediate/About-Different-Middleware-Vendors.html, 2024. ROS 2 Documentation. GitHub: ros2/ros2_documentation, commit 5cdc08c88a3abfa44a8424d6b466a12ebbb7253e. Available at https://github.com/ros2/ros2_documentation/blob/5cdc08c88a3abfa44a8424d6b466a12ebbb7253e/source/Concepts/Intermediate/About-Different-Middleware-Vendors.rst.

[43] Katherine Scott, Chris Lalancette, and Audrow Nash. 2021 ROS Middleware Evaluation Report. https://osrf.github.io/TSC-RMW-Reports/humble/, October 2021. ROS 2 Middleware evaluation reports for each ROS release. GitHub: osrf/TSC-RMW-Reports, Available at https://github.com/osrf/TSC-RMW-Reports.

[44] eProsima. 2022 Fast DDS Performance Testing. https://eprosima.com/index.php/company-all/news/298-2022-fast-dds-performance-testing, December 2022. eProsima's performance testing of Fast DDS, focusing on latency and throughput, compared with other DDS implementations. Location: Madrid, December 2022.

[45] eProsima. Fast DDS vs Cyclone DDS: Benchmarking DDS Implementations. https://eprosima.com/index.php/resources-all/performance/fast-dds-vs-cyclone-dds-performance, December 2022. eProsima presents a comparison between Fast DDS and Eclipse Cyclone DDS, focusing on latency and throughput performance. Updated: 21st December 2022.

[46] eProsima. Fast DDS vs OpenDDS: Benchmarking DDS Implementations. https://eprosima.com/index.php/resources-all/performance/fast-dds-vs-opendds-performance, December 2022. eProsima presents a comparison between Fast DDS and OpenDDS, focusing on latency and throughput performance. Updated: 21st December 2022.

[47] Open Robotics. ROS 2 Alternative Middleware Report. Technical report, Open Robotics, September 2023. Accessed on 2024-03-25.

[48] Phani Gangula. ROS-2 communication optimisation using Zenoh ROS2 Bridge. Presentation at ROSConDE 2023, November 2023. Senior Solutions Architect, Zettascale. Email: Phani@zettascale.tech.

[49] Julien Enoch and Yadunund Vijay. From zero to deployment, ros 2 and zenoh. Webinar presentation slides, 2024. Accessed: September 9, 2024.

[50] Zenoh. What is Zenoh? https://zenoh.io/docs/overview/what-is-zenoh/, 2024. Zenoh Documentation. GitHub: atolab/zenoh-web, commit 9b62ef9cf39894e50dc9730dd7d9ba1a7bd8a7ec. Available at https://github.com/atolab/zenoh-web/blob/master/content/docs/overview/what-is-zenoh.md.

[51] Eclipse Zenoh Developers. Zenoh Plugin for ROS 2 DDS. https://github.com/eclipse-zenoh/zenoh-plugin-ros2dds, 2024. Accessed: September 2024.

[52] Eclipse Zenoh Team. Integrating ROS2 with Eclipse Zenoh. https://zenoh.io/blog/2021-04-28-ros2-integration/, 2024. Zenoh Blog. Available at https://zenoh.io/blog/2021-04-28-ros2-integration/.

[53] rmw_zenoh Developers. ROS 2 RMW Implementation using Zenoh. https://github.com/ros2/rmw_zenoh, 2024. Accessed: September 2024.

[54] Loïck Pierre Chovet, Gabriel Manuel Garcia, Abhishek Bera, Antoine Richard, Kazuya Yoshida, and Miguel Angel Olivares-Mendez. Performance comparison of ros2 middlewares for multi-robot mesh networks in planetary exploration, 2024.

[55] RobotWebTools. rosbridge_suite: Server implementations of the rosbridge v2 protocol. https://github.com/RobotWebTools/rosbridge_suite, 2024. Accessed: 2024-03-19.

[56] Object Computing, Inc. *OpenDDS Developer's Guide*. Object Computing, Inc., December 2022. Accessed: [Insert access date here].

[57] OpenDDS Developers. Internet-Enabled RTPS. https://opendds.readthedocs.io/en/latest-release/devguide/internet_enabled_rtps.html, 2024. OpenDDS Documentation. GitHub: OpenDDS/OpenDDS, commit 83ab8c139083aa47eb7d55ec664b27742c254a45. Available at https://github.com/OpenDDS/OpenDDS/blob/83ab8c139083aa47eb7d55ec664b27742c254a45/docs/devguide/internet_enabled_rtps.rst.

[58] RobotWebTools. webrtc_ros: Streaming of ROS Image Topics using WebRTC. https://github.com/RobotWebTools/webrtc_ros, 2024. GitHub repository for webrtc_ros, offering WebRTC-based streaming solutions for ROS image topics. Last accessed: September 2024.

[59] Quanyan Zhu, Stefan Rass, Bernhard Dieber, Víctor Mayoral Vilches, et al. Cybersecurity in robotics: Challenges, quantitative modeling, and practice. *Foundations and Trends® in Robotics*, 9(1):1–129, 2021.

[60] Quanyan Zhu, Stefan Rass, Bernhard Dieber, and Vıctor Mayoral Vilches. An introduction to robot system cybersecurity. *arXiv preprint arXiv:2103.05789*, 2021.

[61] Justin Wilson. Interoperable internet-enabled dds applications. https://objectcomputing.com/resources/publications/mnb/2019/06/20/interoperable-internet-enabled-dds-applications, 2019. Accessed: Septem-

ber 2024.

[62] Richa Varma, Chris Melville, Claudio Pinello, and Tuhin Sahai. Post quantum secure command and control of mobile agents inserting quantum-resistant encryption schemes in the secure robot operating system. *International Journal of Semantic Computing*, 15(03):359–379, 2021.

[63] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and Scott Simon. NIST SP 800-56B Rev. 2. Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography, 2019.

[64] NIST. FIPS 186-5. Digital Signature Standard (DSS), 2023.

[65] National Institute of Standards and Technology. Draft fips (federal information processing standards) 203, module-lattice-based key-encapsulation mechanism standard. Technical report, NIST, Information Technology Laboratory, 2003.

[66] National Institute of Standards and Technology. Draft fips (federal information processing standards) 204, module-lattice-based digital signature standard. Technical report, NIST, Information Technology Laboratory, 2003.

[67] National Institute of Standards and Technology. Draft fips (federal information processing standards) 205, stateless hash-based digital signature standard. Technical report, NIST, Information Technology Laboratory, 2003.

[68] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246, IETF, Aug 2008.

[69] E. Rescorla and N. Modadugu. Datagram transport layer security version 1.2. RFC 6347, IETF, Jan 2012.

[70] Eric Rescorla. The transport layer security (tls) protocol version 1.3. RFC 8446, IETF, 2018.

[71] Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. The datagram transport layer security (dtls) protocol version 1.3. RFC 9147, IETF, Apr 2022. Obsoletes RFC 6347.

[72] Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE, 2015.

[73] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 841–852, 2020.

[74] Thom Wiggers. *Post-Quantum TLS*. Ph.d. thesis, Radboud University, 2024.

[75] Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in tls 1.3. Technical report, Internet-Draft, Internet Engineering Task Force (IETF) Network Working Group, Sep 2023. Draft-Ietf-Tls-Hybrid-Design-09.

[76] Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. In *Selected Areas in Cryptography (SAC) 2016*, volume 10532 of *Lecture Notes in Computer Science*, pages 1–24. Springer, oct 2017.

[77] Open Quantum Safe Project. liboqs: Library for quantum-resistant cryptographic algorithms. https://github.com/open-quantum-safe/liboqs, 2024. Accessed: 2024-03-21.

[78] Open Quantum Safe Project. OpenSSL 3 provider containing post-quantum algorithms. https://github.com/open-quantum-safe/oqs-provider, 2024. Accessed: 2024-03-21.

[79] wolfSSL. DTLS 1.3 support for Post-Quantum Cryptography. https://www.wolfssl.com/dtls-1-3-support-post-quantum-cryptography/, January 2023. Do you want to start using wolfSSL's DTLS 1.3 implementation?

[80] Christian Esposito and Mario Ciampi. On security in publish/subscribe services: A survey. *IEEE Communications Surveys & Tutorials*, 17(2):966–997, 2014.

[81] Thomas White, Michael N Johnstone, and Matthew Peacock. An investigation into some security issues in the dds messaging protocol. 2017.

[82] eProsima. *6.7. TLS over TCP — Fast DDS 2.13.1 documentation*, 2023. Accessed: 2024-02-15.

[83] Real-Time Innovations, Inc. RTI TLS Support Release Notes, Version 7.2.0. Technical report, Real-Time Innovations, Inc., October 2023. © 2010-2023 Real-Time Innovations, Inc. All rights reserved.

[84] Real-Time Innovations. *RTI Security Plugins User's Manual: Choosing the Right Technology to Protect Your Data*. Accessed: 6 march 2024.

[85] Hamed Soroush, David Arney, and Julian Goldman. Toward a safe and secure medical internet of things. *IIC J. Innov*, 2(1):4–18, 2016.

[86] DDS Security Version 1.1. https://www.omg.org/spec/DDS-SECURITY/1.1/About-DDS-SECURITY, July 2018. OMG Document Number: formal/2018-04-01.

[87] Javier Blanco-Romero, Vicente Lorenzo, Florina Almenares, Daniel Díaz Sánchez, and Adrián Serrano Navarro. Pqsec-dds: Integrating post-quantum cryptography into dds security for robotic applications. *Jornadas Nacionales de Investigación en Ciberseguridad (JNIC)(9ª. 2024. Sevilla)(2024), pp. 396-403.*, 2024.

[88] Eclipse Zenoh. TLS authentication. https://zenoh.io/docs/manual/tls/, 2024. Zenoh Documentation. GitHub: atolab/zenoh-web, commit ed0fdc49e19c6d8cd27c3fa92741a9f09a39e741. Available at https://github.com/atolab/zenoh-web/blob/ed0fdc49e19c6d8cd27c3fa92741a9f09a39e741/content/docs/manual/tls.md.

[89] Eclipse Zenoh. Zenoh charmander grows stronger. https://zenoh.io/blog/2023-06-05-charmander2/, 2023. Zenoh Blog. Accessed on 2024-03-25.

[90] Eclipse Zenoh. QUIC transport. https://zenoh.io/docs/manual/quic/, 2024. Zenoh Documentation. GitHub: atolab/zenoh-

web, commit 842232f6ef125df4f4346b4edaf2bb55b8d99df8. Available at https://github.com/atolab/zenoh-web/blob/master/content/docs/manual/quic.md.

[91] rustls Developers. Post-Quantum Provider in rustls. https://github.com/rustls/rustls/tree/main/rustls-post-quantum, 2024. GitHub: rustls/rustls, branch main, last accessed September 2024. Available at https://github.com/rustls/rustls/tree/main/rustls-post-quantum.

[92] rustls Developers. rustls-post-quantum Documentation. https://docs.rs/rustls-post-quantum/latest/rustls_post_quantum/, 2024. Documentation for rustls-post-quantum crate, version 0.1.0. Last accessed September 2024. Available at https://docs.rs/rustls-post-quantum/latest/rustls_post_quantum/.

[93] Bas Westerbaan and Douglas Stebila. X25519Kyber768Draft00 Hybrid Post-Quantum Key Agreement. Internet-Draft draft-tls-westerbaan-xyber768d00-03, Internet Engineering Task Force (IETF), September 2023. Expired and archived. Last updated March 2024. Available at https://datatracker.ietf.org/doc/draft-tls-westerbaan-xyber768d00/03/.

[94] Kyle Fazzari. ROS 2 DDS-Security integration. https://design.ros2.org/articles/ros2_dds_security.html, Jul 2019. ROS 2 Design Article. GitHub: ros2/design, commit 12f61b14698b80170824c699c70608d9ded3a6d7. Available at https://github.com/ros2/design/blob/12f61b14698b80170824c699c70608d9ded3a6d7/articles/180_ros2_dds_security.md.

[95] Open Robotics. Setting up security. https://docs.ros.org/en/jazzy/Tutorials/Advanced/Security/Introducing-ros2-security.html, 2024. ROS 2 Documentation. GitHub: ros2/ros2_documentation, commit 0684f6b32045113b5a16496c78f2b49d8bce452d. Available at https://github.com/ros2/ros2_documentation/blob/0684f6b32045113b5a16496c78f2b49d8bce452d/source/Tutorials/Advanced/Security/Introducing-ros2-security.rst.

[96] Javier Blanco-Romero. Sros2 with post-quantum cryptography support. https://github.com/fj-blanco/sros2, 2024.

[97] Javier Blanco-Romero. Zenoh PQ branch. https://github.com/fj-blanco/zenoh/tree/pq, 2024. GitHub repository branch for post-quantum cryptography enhancements in Zenoh.

[98] SROS2 Developers. Sros2 github repository. https://github.com/ros-sros2/sros2, 2023. Accessed: October 2023.

[99] Python Cryptographic Authority. cryptography, 2024.

[100] Open Quantum Safe. S/MIME message signing – Cryptographic Message Syntax (CMS). https://github.com/open-quantum-safe/oqs-provider/blob/c9b8056cbe3bba1a6c151b0a8af6a15b8ce32fb6/USAGE.md#smime-message-signing----cryptographic-message-syntax-cms, 2024. OQS Provider Documentation. GitHub: open-quantum-safe/oqs-provider, commit c9b8056cbe3bba1a6c151b0a8af6a15b8ce32fb6. Avail-

able at https://github.com/open-quantum-safe/oqs-provider/blob/c9b8056cbe3bba1a6c151b0a8af6a15b8ce32fb6/USAGE.md.