國立臺北科技大學
National Taipei University of Technology

資訊安全碩士學位學程

碩士學位論文

Master of Science in Information Security

Master Thesis

即時醫療資訊傳輸中 DDS 安全性擴充功能之性能探討與評估

Evaluation of DDS Security Plugin Capabilities in Real-Time Healthcare Data Transmission

研究生：鄧仔君

Researcher: Yu-Jun Deng

指導教授：陳香君 博士

Advisor: Shiang-Jiun Chen, Ph.D.

July 2025

國立臺北科技大學
研究所碩士學位論文口試委員會審定書

本校 資訊安全碩士學位學程 研究所 鄧仔君 君

所提論文，經本委員會審定通過，合於碩士資格，特此證明。

學位考試委員會

委　　員：吳和庭

馬奕薇

陳香君

指導教授：陳香君

所　　長：

中 華 民 國　一百一十四　年　七　月　九　日

ABSTRACT

Keywords: Data Distributed Service (DDS), Security Plugin, RTI Connext Secure, medical

Data Distribution Service (DDS) is a middleware protocol for real-time distributed systems and is widely applied across various domains. In the medical field, it enables efficient and scalable communication. The DDS Security Plugin Interface (SPI) further strengthens protection through configurable Quality of Service (QoS) policies.

However, integrating DDS into real-time medical systems remains complex. Medical environments demand data protection and low-latency communication requirements, making this integration challenging. While the SPI framework supports flexible implementations, the configuration options provided by vendors can be complex and error-prone, this also increases the risk of misconfiguration, which may lead to system vulnerabilities. Moreover, existing research on this topic remains limited. Comprehensive studies evaluating how Security Plugins protects real-time medical data are still lacking.

To address this gap, this study simulates a real-time medical communication environment based on RTI Connext Secure and the MetaCares Monitoring Station developed by MetaCares Intelligent Inc. Several scenarios are designed to reflect potential security threats encountered in practice, aiming to evaluate the capabilities of Security Plugins. Experimental results show that security plugins can mitigate threats such as unauthorized access to communication domains and exposure or tampering of sensitive medical data, while maintaining performance overhead below 0.1 seconds in average latency increase and no significant performance degradation. This study provides empirical evidence for the practical application of DDS Security Plugins in healthcare environments and contributes to a better understanding of the interplay between security configurations and system performance in real-time medical data exchange.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Shiang-Jiun Chen, whose patient guidance and steadfast support were instrumental throughout the course of this research. From the early stages of conceptual development to the final revisions of this thesis, Professor Chen's insightful feedback, professional expertise, and unwavering encouragement were a constant source of motivation. I am truly grateful for the opportunity to learn under her mentorship. I am also deeply grateful to the management and engineers of MetaCares Intelligent Inc. and YCL Technology, Inc. for their generous support throughout this study. I am especially grateful for their permission to use the MetaCares Monitoring Station and related software resources, which formed the foundation for the experimental environment. Their assistance and cooperation were invaluable to the progress and completion of this research.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

The rapid advancement of 5G technology has led to an unprecedented demand for low latency and higher transmission speeds, fundamentally transforming real-time data communication. In this context, Data Distribution Service (DDS) [1], an open international middleware standard published by the Object Management Group (OMG), presents a viable solution for distributed systems. It employs a publish-subscribe communication model and a decentralized architecture, enabling high throughput and low latency. With configurable Quality of Service (QoS) policies, DDS supports precise tuning of reliability and resource utilization, making it particularly suitable for time-sensitive applications [2].

The increasing demand for real-time, interoperable systems has fueled the growth of DDS across various industries [3]. According to market research, the global market for this middleware was valued at USD 104.21 million in 2023 and is expected to grow to USD 446.47 million by 2029, with a compound annual growth rate (CAGR) of 27.44% [4]. This rapid expansion highlights its growing role in mission-critical applications. In the medical domain in particular, the standard has enabled seamless communication between devices, supporting accurate and real-time transmission of patient data, which is essential for effective clinical decision-making and patient monitoring [5]. To enhance system protection and ensure trusted data exchange, DDS supports standardized Security Plugin Interfaces (SPIs), whose behavior can be configured through QoS [6].

However, integrating these SPIs into real-time medical systems remains a complex task. The implemented security plugins implementing these interfaces require careful tuning to balance protection and system performance. This balance is particularly challenging in medical environments, where strict data protection and low-latency communication are essential. While the Security Plugins can be configured flexibly, this flexibility may result in security vulnerabilities because of misconfiguration. As reported in [7], 95% of tested applications had at least one misapplied configuration, underscoring the widespread impact of deployment issues. Similar concerns apply to DDS-based systems, where improper setup may undermine the intended protection. Despite these risks, current research on the practical use and configuration of security plugins in medical communication environments remains limited.

Therefore, the goal of this paper is to explore the influence of Security Plugin configuration on data security and transmission efficiency in real-time medical environments.

This includes how to identify the capabilities of DDS security plugins in medical systems, with a focus on achieving a balance between data protection and transmission performance. To achieve this, the study is conducted in a simulated ICU system that incorporates the MetaCares Monitoring Station developed by MetaCares Intelligent Inc. and RTI Connext Secure. Within this system, multiple test scenarios are designed based on realistic challenges in medical settings. Different security plugins, such as authentication, access control, and encryption, are then applied to observe how these settings defend against potential risks while managing system resources.

The experimental results show that enabling security plugins effectively prevents unauthorized access and data leakage in the simulated threat scenarios, with an average latency increase of approximately 0.1 seconds and a 0.1% rise in memory usage. These results demonstrate that appropriate security plugins can enhance data protection while maintaining the low latency and high reliability required for medical data transmission.

The main contributions of this study are as follows:

1. Systematic evaluation of Security Plugins in a simulated DDS-based ICU system.
2. Analyze the impact of different security plugin configurations on system security and performance.
3. Providing practical guidelines for secure DDS deployment in healthcare.
4. Addresses a research gap regarding the trade-offs between security and transmission efficiency in medical data communication with DDS security plugins.

The remainder of this paper is structured as follows: Chapter 2 reviews related work, providing an overview of DDS and its security mechanisms. Next, QoS capabilities and performance evaluations are reviewed. Additionally, the chapter discusses RTI Connext Secure, providing context for the study's experiments. Chapter 3 presents the research methodology, covering experimental design, including system architecture and scenarios. Then, the simulated ICU system deployment and the process used to evaluate the impact of the security plugins on security and performance are described. Chapter 4 presents experimental results, analyzing how the security plugin capabilities respond under different scenarios, with a focus on their impact on system behavior, security, and performance. Finally, Chapter 5 concludes the research with a summary, discussing the security plugin implications, and suggesting directions for future research in optimizing DDS-based medical data transmission.

Chapter 2 Related Work

This section reviews key literature relevant to this study. It covers DDS fundamentals, including its architecture, applications in healthcare, and security mechanisms. It also reviews prior evaluations of QoS-related performance in DDS-based systems, highlighting how different configurations affect metrics such as latency, reliability, and throughput. A detailed discussion of RTI Connext Secure follows, focusing on its integration of the DDS Security standard and the current lack of literature exploring its security capabilities. These discussions provide the foundational context for understanding the basic concepts of DDS-based systems and the role of DDS in medical data communication. The insights gained from this literature review highlight a gap in in-depth studies on secure plugin configuration and inform the experimental approach presented in the next chapter.

2.1 Data Distributed Service

2.1.1 Overview

Data Distribution Service (DDS) is an open middleware standard that aims to achieve low latency and high performance in a real-time system. Unlike traditional message-oriented middleware such as Message Queuing Telemetry Transport (MQTT) and Advanced Message Queuing Protocol (AMQP), which rely on discrete message exchanges, DDS adopts a data-centric approach. This means that the middleware maintains awareness of the data it stores and controls how the data is shared across the network [8]. By structuring communication around shared data, DDS minimizes direct coupling between senders and receivers, enhancing dynamic scalability and reducing network congestion. These features make DDS especially suitable for distributed real-time applications [9].

Architecturally, DDS defines two interface layers: the Data Local Reconstruction Layer (DLRL) [10] and the Data-Centric Publish-Subscribe (DCPS) layer. The emphasis below is on the latter, which serves as the core of DDS and is responsible for data publishing and subscribing, management, and transmission. DDS standard employs a publish-subscribe communication model within a decentralized, peer-to-peer communication system. This allows nodes to dynamically join or leave the network. Figure 1 illustrates the key architectural components of DDS involved in data transmission, including the roles of Participant, Publisher, Subscriber, DataWriter, and DataReader. These entities operate within the same

domain to enable efficient and decoupled communication.

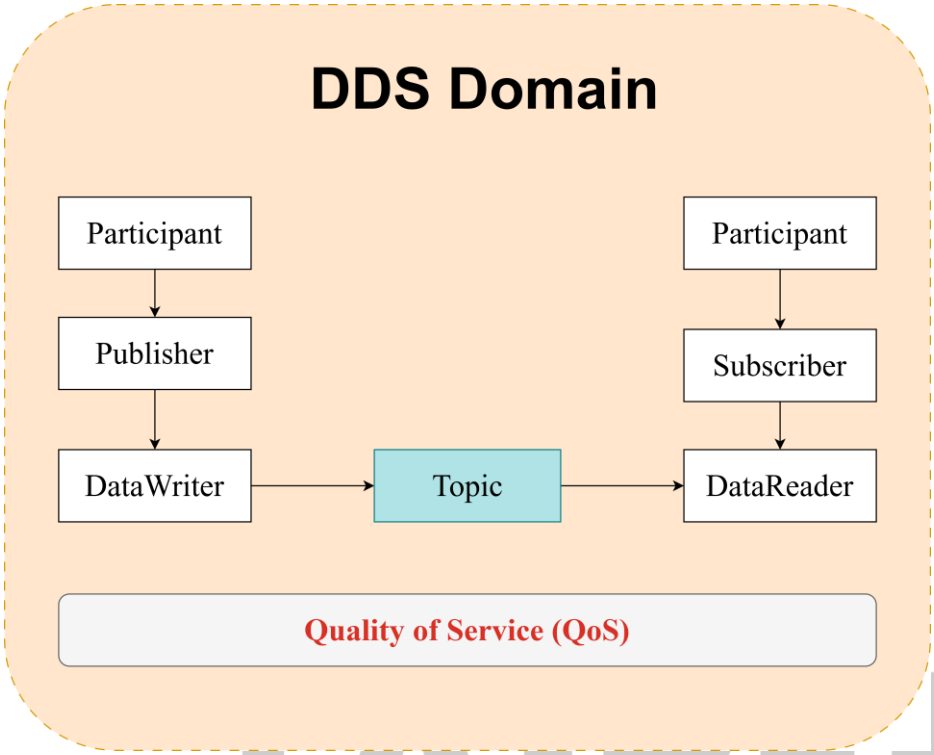


Figure 1. Architecture of Data Distribution Service (adapted from [11]).

At the core of DDS is the concept of a Domain, which defines an isolated communication space where only participants within the same domain can exchange data. Within each domain, the Participant acts as a container for DDS entities. It is the primary interface for applications interacting with the DDS domain, managing communication entities such as publishers, subscribers, topics, and QoS settings. Publishers are responsible for managing DataWriters, which send data to specific topics. Conversely, subscribers manage DataReaders, which receive data from those topics. Topic acts as an abstraction layer and defines the type of data to be communicated. It allows multiple publishers and subscribers to interact with each other without requiring direct knowledge, thereby promoting loose coupling. Thus, communication occurs indirectly through topics, decoupling data producers and consumers. Communication is achieved exclusively through topics, allowing multiple publishers and subscribers to participate in data exchange in a flexible and scalable manner.

Data transmission in DDS begins with the discovery phase, during which newly joined Participants automatically detect and establish connections with other participants in the same

domain. This auto-discovery mechanism eliminates the need for manual configuration and enables dynamic system expansion. Once discovery is complete, a DataWriter publishes structured data to a topic, which is then made available to any matching DataReader instances. This entire process is governed by QoS policy, which defines how data is sent, stored, and delivered. For instance, the Reliability policy determines whether data should be delivered reliably or on a best-effort basis. Other QoS settings specify whether historical data should be retained for late-joining subscribers or whether data should persist after the publisher disconnects.

Several studies have highlighted the practical advantages of DDS in real-time environments, especially in terms of scalability, efficiency, and interoperability. For example, Jawad Ali et al. [8] mentioned that DDS provides an interoperable and scalable framework for real-time, secure data transactions, making it well-suited for IoT and machine-to-machine communication. Xiaowen, Z et al. [2] demonstrated that a DDS-based framework enables low-latency and high-reliability remote procedure calls in industrial robotics by leveraging a topic-centric request-response mechanism and dynamic service discovery. Ioana, A. et al. [9] emphasized that DDS's flexibility stems from its dual support for publish-subscribe and request-reply paradigms, enabling efficient data exchange in complex systems and meeting evolving IIoT demands.

2.1.2 Applications of DDS in Healthcare

DDS has been widely adopted in healthcare applications due to its ability to provide real-time, reliable, and scalable data exchange. This section presents representative use cases of DDS in the medical field.

For instance, the integration of NVIDIA Holoscan with RTI Connext DDS enables AI-powered medical devices by facilitating low-latency, high-throughput data transmission in clinical settings [12]. DDS is also used in patient monitoring systems by facilitating real-time communication between medical devices. This improves interoperability and supports coordinated operation across various patient monitors and electronic health record systems [15]. Another significant initiative is the MD PnP Interoperability Initiative, supported by NIST, which explores DDS for seamless device connectivity [16]. Collectively, DDS enhances real-time data exchange, interoperability, and imaging workflows, making it an important technology in modern healthcare.

2.1.3 Security Mechanisms

Although DDS was originally designed for efficient real-time communication, it lacked built-in security. As the demand for security in distributed systems has increased, OMG introduced the DDS Security specification in 2004 to address potential vulnerabilities of DDS in open environments. Since then, the standard has undergone multiple revisions, with version 1.2 released in 2024 [17]. This specification defines five SPIs that collectively ensure data transmission confidentiality, integrity, and availability in DDS-based systems. It primarily covers authentication, access control, encryption, message integrity, and logging.

2.1.3.1 Authentication Mechanism

The authentication mechanism in DDS Security ensures that only authorized participants can join the data distribution domain, thereby preventing impersonation and unauthorized access [18]. At its core, the mechanism relies on Public Key Infrastructure (PKI) and employs X.509 certificates for identity verification. When a participant attempts to join a DDS domain, it must present valid credentials, such as a certificate. Upon successful verification, the participant is granted access and issued a shared secret, which is subsequently used to establish communication. This structured authentication process not only confirms identity but also mitigates risks associated with rogue nodes attempting to impersonate legitimate participants. Enforcing strict authentication policies and leveraging cryptographic techniques, the authentication module establishes a secure communication environment in which only verified entities are permitted to exchange data. This security measure is crucial in scenarios where unauthorized access could result in serious consequences [18]. To enhance this process, a recent patent [20] introduces a systematic approach for synchronizing encryption certificates among participants, enhancing efficiency in key management. Overall, the authentication mechanism plays a fundamental role in maintaining the confidentiality and integrity of data exchanged within the distributed system.

2.1.3.2 Access Control Mechanism

The Access Control mechanism in DDS Security governs what actions an authenticated participant is permitted to perform, such as publishing, subscribing, or interacting with specific DDS topics [18]. This ensures that participants operate within predefined security policies to prevent unauthorized data access or modifications. At the core of this mechanism lies a policy-based approach, configured through two main XML-based

files: the governance file and the permissions file [18]. The governance file defines the global security posture of the system, such as whether authentication is required, whether encryption is enforced, and which permissions file to apply. The permissions file then specifies detailed access rules for each participant, supporting role-based access control (RBAC) and enabling fine-grained authorization at the topic or instance level. In addition to controlling domain access, the access control logic also applies to publishing, subscribing, and even writing or reading specific topic instances [18].

To maintain integrity and prevent tampering, all permissions are cryptographically signed. The access control evaluation occurs immediately after authentication. If the requested action aligns with the participant's role and policy, access is granted. Otherwise, unauthorized operations, such as publishing to a restricted topic, are strictly denied. This mechanism is especially crucial in high-security environments, ensuring that only permitted entities interact with sensitive data while maintaining data confidentiality and integrity [19].

2.1.3.3 Encryption Mechanism

The Cryptographic mechanism in DDS Security ensures the confidentiality, integrity, and authenticity of DDS communications by handling encryption, message authentication, and digital signatures [18]. This protects against unauthorized access, data tampering, and various security threats, such as eavesdropping and replay attacks. At its core, the mechanism encrypts both DDS messages and metadata, ensuring that only authorized participants can access transmitted information. To detect and prevent tampering, it employs message authentication codes (MACs), which validate data integrity. It supports multiple cryptographic algorithms, including AES-GCM for authenticated encryption, HMAC-SHA256 for message authentication, and Elliptic Curve Cryptography (ECC) for digital signatures. These algorithms provide a robust defense against attacks like man-in-the-middle (MITM) intrusions [19]. The encryption and authentication workflow follows a structured process: before transmission, a DataWriter encrypts its messages using a shared key and generates a MAC to ensure integrity. Upon reception, the DataReader decrypts the message using the appropriate key and verifies its integrity using the MAC. This mechanism guarantees that only legitimate participants can read and validate DDS data, reinforcing secure communication in mission-critical systems [18]. The importance of this mechanism is further underscored by recent security models and patent disclosures, which highlight the importance of systematic encryption certificate synchronization and secure key exchange among participants [20].

2.1.4.4 Logging and Tagging Mechanism

The Logging and Data Tagging mechanisms are important components of DDS Security, designed to enhance transparency and enforce fine-grained data protection. While they are not always the primary focus in real-time performance evaluations, these SPIs contribute significantly to the overall security posture of DDS-based systems [18-19].

The Logging mechanism serves as the system's audit layer by recording security-relevant events to support compliance verification, debugging, and forensic analysis. It systematically logs activities such as authentication failures, unauthorized access attempts, and encryption errors. These records can be integrated into external monitoring systems for centralized, real-time oversight. The Tagging mechanism provides fine-grained classification of data by allowing each DataWriter to assign security labels such as "Confidential" or "Internal Use Only" to individual data samples. These tags convey information about the sensitivity level and the data origin. The Access Control mechanism uses these tags to dynamically enforce access policies based on both participant identity and data classification. This dynamic enforcement is vital in sensitive domains, including healthcare and defense, where protecting data confidentiality and integrity is paramount [18-19].

Together, these mechanisms strengthen the auditability and classification capabilities of DDS Security, complementing the foundational protections offered by authentication, access control, and encryption.

2.2 Performance Evaluation of QoS Configurations in DDS

Quality of Service (QoS) policies are fundamental to how DDS delivers real-time, reliable communication. These policies allow developers to control aspects such as reliability, latency, data lifespan, and participant availability. For instance, the Reliability policy determines whether data should be delivered reliably or on a best-effort basis, while the Latency Budget and Deadline policies define acceptable transmission delays and update frequencies. Together, these settings allow DDS to adapt to a wide range of application needs from bandwidth-sensitive systems to mission-critical real-time environments.

In practice, selecting appropriate QoS policies can significantly affect system performance. Several studies have discussed how these settings influence metrics such as latency, throughput, and reliability. For instance, R. S. Auliya et al. [21] proposed an adaptive algorithm that achieved communication reliability ranging from 99% to 99.99% across various settings, while slightly improving per-topic throughput. Similarly, Zhuangwei Kang

et al. [22] evaluated the performance of DDS, MQTT, and ZeroMQ in IoT applications under varying load conditions and QoS configurations. Their results indicated that DDS offers the most comprehensive QoS support and achieves superior reliability, with “Multicast”, “TurboMode”, and “AutoThrottle” contributing to reduced latency and increased throughput under varying message sizes and publication rates. These results underline the importance of tuning QoS parameters to maintain optimal communication performance.

In addition to typical network scenarios, DDS’s QoS policies have proven effective and consistent in heterogeneous and time-sensitive networks. Takrouni et al. [23] explored the integration of DDS into vehicular networks with a model-based Simulink approach. Their study also validates QoS performance across FlexRay, Gigabit Ethernet, and Avionics Full-Duplex Switched Ethernet (AFDX) networks. Their findings indicated that DDS could maintain stable QoS behavior and consistent performance across different network architectures, showcasing the middleware’s adaptability in complex environments.

While QoS policies offer significant flexibility, their complexity introduces configuration challenges that may impact system stability and performance. A. Alaerjan [24] formalized the semantics of DDS QoS policies using Object Constraint Language (OCL) and analyzed their interdependencies. This approach aids in preventing misconfigurations that could degrade system performance, highlighting the need for careful QoS management in DDS-based applications.

Beyond configuration complexity, the performance of DDS systems under QoS policies also depends on the specific DDS implementations. Kaleem Peeroo et al. [25] conducted a systematic survey of experimental performance evaluations across multiple DDS implementations, analyzing how different QoS configurations and non-QoS factors affect latency, throughput, and jitter. They compared DDS implementations against each other and alternative communication technologies, examined scalability effects, and identified gaps in existing DDS performance research. Complementarily, Vincent Bode et al. [26] developed DDS-Perf, a cross-vendor benchmarking tool, and used it to analyze the performance of four DDS implementations, finding that RTI Connext excelled in overall performance while FastDDS had the lowest end-to-end latency. Their result indicates that even with identical QoS policies, implementation details significantly influence overall performance.

Collectively, these studies demonstrate that QoS policies not only serve as a powerful mechanism for optimizing DDS communication but also introduce configuration complexity that may affect system performance. Commonly adopted performance metrics across these

studies include latency and throughput, which are consistently used to evaluate the efficiency and responsiveness of DDS-based communication under various configurations and network conditions. These metrics form a practical basis for understanding the performance implications of different QoS strategies.

2.3 RTI Connext Secure

RTI Connext Secure [18] is a secure middleware solution based on RTI Connext DDS, which is a widely used implementation of the DDS standard. It integrates the DDS Security standard through a plugin-based architecture, ensuring data confidentiality, integrity, and access control. Although RTI Connext DDS's performance and QoS features have been extensively studied, research focusing on the security aspects and their impact on system performance remains limited. Most existing work concentrates on core DDS functionalities and QoS tuning without deeply analyzing the overhead introduced by security mechanisms.

The security plugins in RTI Connext Secure are implemented through the DDS SPI, which provides a modular framework for integrating essential security functions into the DDS middleware. This SPI defines standardized interfaces for components such as Authentication, Access Control, Cryptographic, and Logging plugins. By leveraging this architecture, RTI Connext Secure enables these security plugins to be developed, configured, and updated independently, ensuring flexibility and scalability in protecting data exchanges. The SPI-based design facilitates enforcement of participant identity verification, fine-grained permission policies, and encryption of data in transit, all while maintaining interoperability with DDS's real-time communication capabilities.

Despite the growing attention to the security of DDS, studies specifically addressing RTI Connext Secure remain limited, mostly concentrating on core DDS functionalities and QoS configurations. Therefore, this study aims to fill this gap in the literature by exploring the security features of RTI Connext Secure and analyzing their effects on performance in real-time healthcare scenarios, offering a focused contribution to future research.

Chapter 3 Methodology

This chapter outlines the research methodology and system architecture employed in this study, focusing on the application of DDS security plugin capabilities through the configuration of governance and permissions files to balance data security and performance within a simulated ICU system. It begins by detailing the system architecture adopted to investigate the effects of security plugin configurations, and the designed scenarios are presented subsequently. Furthermore, the experiment setup and implementation are described. The content of this chapter builds upon the related work discussed in Chapter 2, providing a foundation for the subsequent analysis and discussion of experimental results in Chapter 4.

3.1 Experimental Design

The research methodology comprises the architecture of the experimental environment and the design of experimental scenarios. These two components together enable the evaluation of how DDS Security configurations affect system safety within a simulated ICU system. Details regarding the experimental environment, scenario design, and plugin configurations are described below.

3.1.1 System Architecture

The system architecture used in this experiment is constructed based on MetaCares Monitoring Station, a real-time monitoring application provided by MetaCares Intelligent Inc., and replicates a typical ICU data communication environment by deploying instances of the MetaCares Monitoring Station across multiple simulated nodes. As shown in Figure 2, the system architecture consists of a nurse station (NS), multiple bedside stations (BS), and simulated ventilators, each connected to a BS. The system also includes a backend database (DB) and a DDS middleware layer responsible for real-time data distribution. BS refers to MetaCares Monitoring Station instances deployed on ICU beds for local monitoring, while NS refers to a centralized instance of the same system used by nurses to observe data from multiple beds. The DB simulates the hospital information center. All components are connected through a network switch to simulate the hospital's internal network infrastructure. The architecture replicates the data flow and communication patterns typical in an ICU system, providing a controlled environment to evaluate how DDS Security configuration can reduce potential risk and maintain system safety and performance.

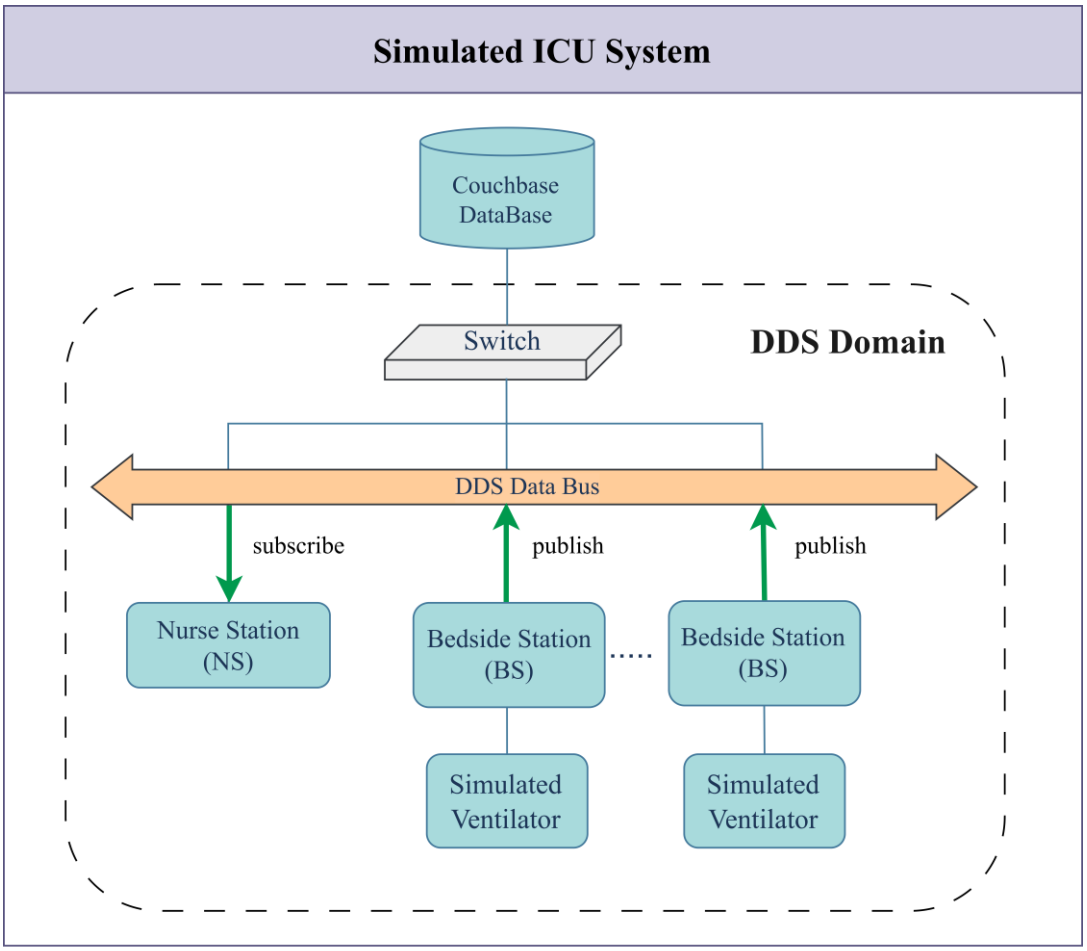


Figure 2. Architecture of the simulated ICU system.

In the architecture, patient data such as respiratory rate and oxygen concentration are generated by simulated ventilators and transformed into predefined DDS topic formats. These data are displayed locally at the corresponding BS, enabling healthcare practitioners to monitor the patient directly at the BS. In addition, the system incorporates an alert mechanism that notifies medical professionals of abnormal physiological conditions, reducing the need for continuous bedside presence and helping to mitigate staffing burdens.

The data are also published to designated DDS topics and subscribed to by the nurse station, which receives information from multiple BS and aggregates data into a unified monitoring dashboard for centralized real-time observation. Meanwhile, the backend database subscribes to the same topics to store the data for future retrieval or analysis. This data

exchange is facilitated by the DDS middleware's publish-subscribe model, which decouples data producers and consumers to support efficient, scalable, and low-latency communication.

3.1.2 Scenario Design

Based on Figure 2, a series of representative scenarios is designed based on potential threats and situations that may arise in real-world medical environments to ensure that the evaluation reflects practical security needs. These scenarios simulate realistic communication and access conditions to evaluate the effectiveness of different security plugin configurations in preventing unauthorized access, data interception, and system compromise. Each case focuses on how the plugins respond to specific security threats.

Scenario 1: An unauthorized participant joins the communication domain

In real-world ICU systems, the presence of multiple interconnected medical devices and network nodes increases the risk of unauthorized systems attempting to join the communication infrastructure. Such incidents may stem from misconfigured prototype devices under development, unregistered personal equipment inadvertently connected to the hospital network, or in more severe cases, malicious actors exploiting weak or absent access controls to gain entry permit into the data domain. Given the sensitivity of physiological data, any unauthorized access to the communication domain poses a significant threat to patient privacy and system integrity. Although this does not directly grant them access to topic-level data, it increases the number of connected nodes within the system, thereby expanding the potential surface for misuse or unauthorized observation, particularly if other access control mechanisms such as read and write policies are not strictly enforced. This scenario is illustrated in Figure 3, where an unauthorized participant attempts to join the DDS Domain.

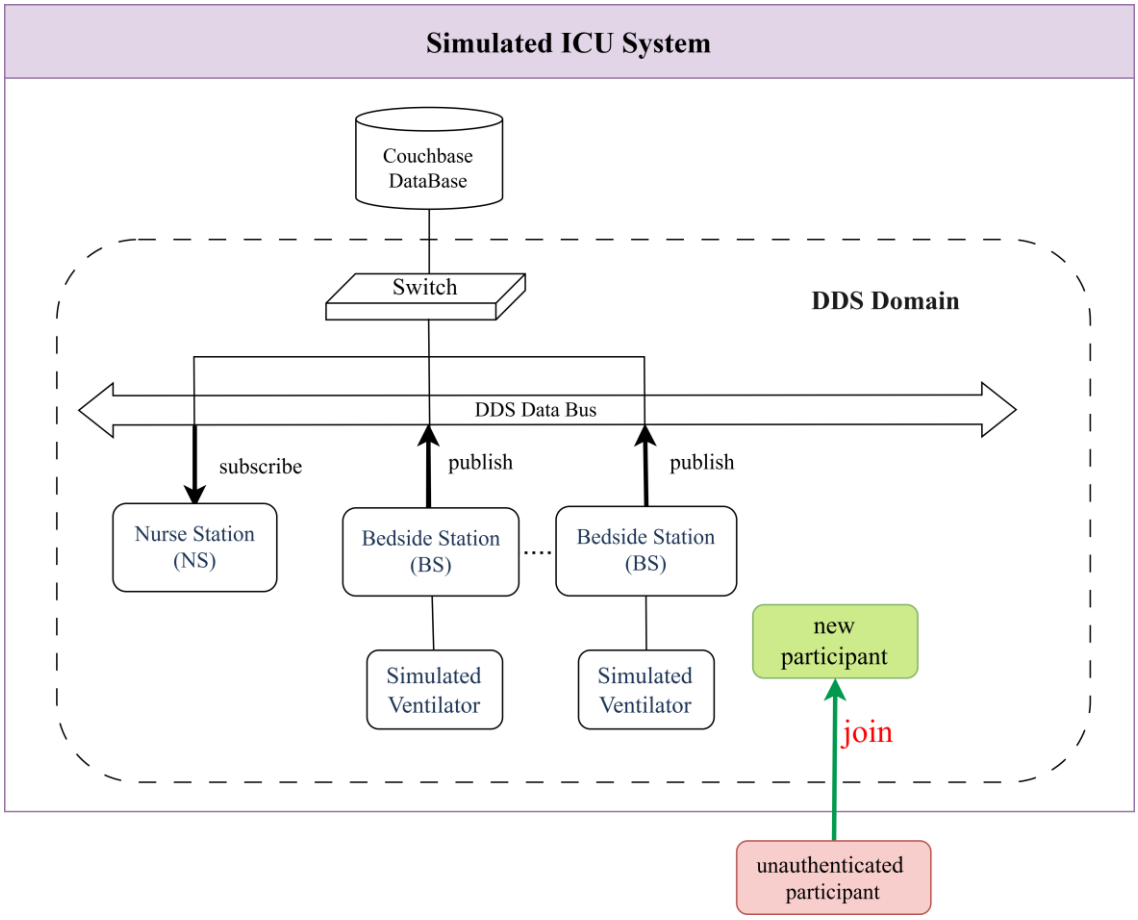


Figure 3. Unauthorized participant attempting to join the DDS domain.

To mitigate this risk, RTI Security Plugins provide configurable rules for participant authentication and domain-level access control, implemented through the pluggable architecture defined in the DDS Security specification. Figure 4 illustrates the decision-making process when a new participant attempts to join the domain. The flow incorporates two critical security plugin rules:

(a) `allow_unauthenticated_participants`, which determines whether identity credentials are required during the discovery phase.

(b) `enable_join_access_control`, which governs whether topic-level permissions are enforced after authentication.

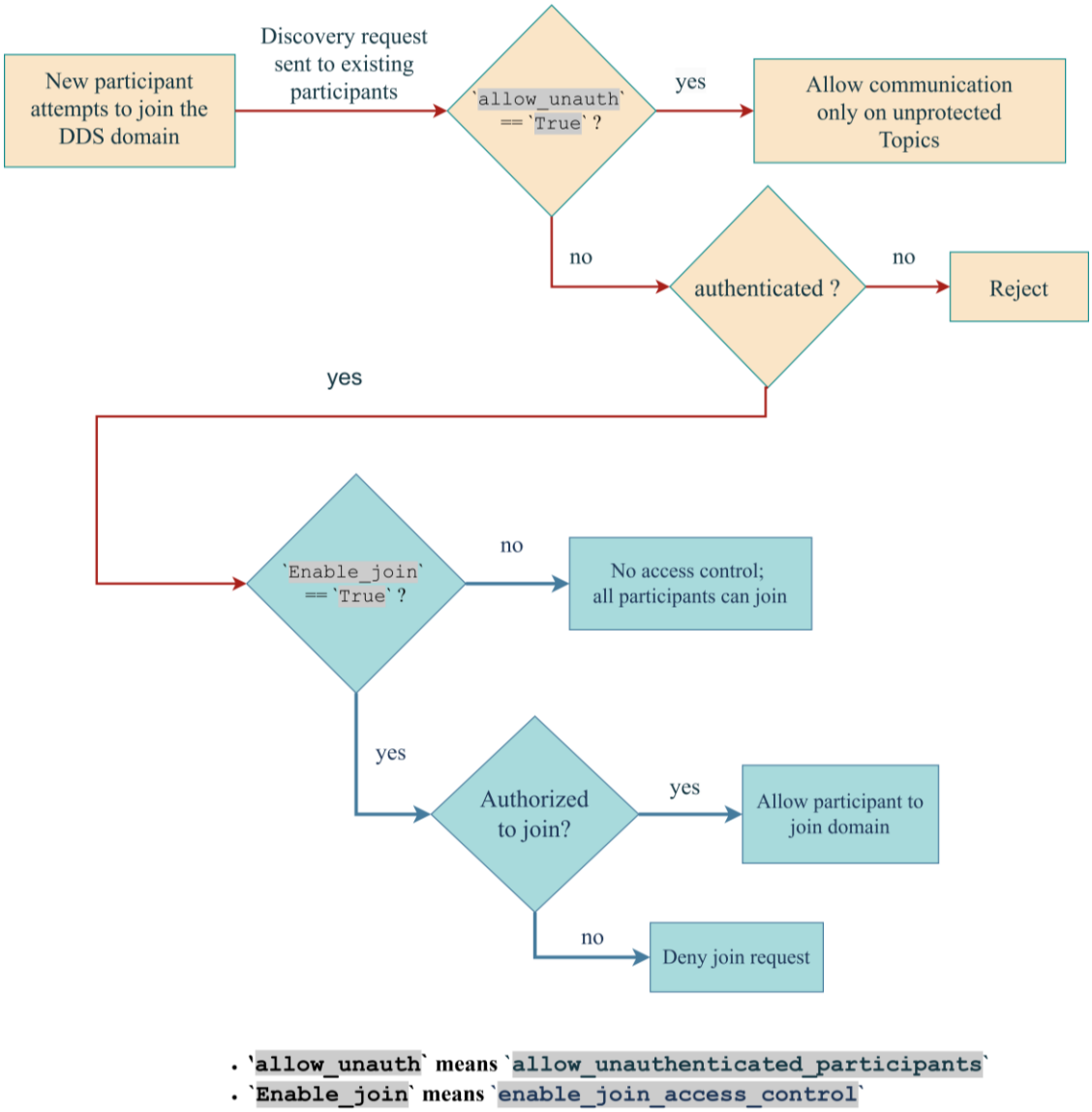


Figure 4. Participant authentication and access control flow under the security plugins.

The flow begins when a new participant attempts to join the domain by sending a discovery request to existing participants. At this point, the system evaluates whether the security plugin setting `allow_unauthenticated_participants` is enabled. This setting is defined in the governance file and determines whether identity credentials are required during the discovery phase. It effectively controls whether unauthenticated participants are permitted to join the network or not.

When the `allow_unauthenticated_participants` setting is set to `True`,

participants lacking valid identity credentials can successfully join the DDS domain. These unauthenticated participants are not subjected to any authentication or subsequent authorization checks because they bypass the entire security validation process. This effectively disables all domain-level and topic-level access control mechanisms. Such a configuration essentially opens the data bus to any node, including untrusted or malicious actors, posing a serious threat to patient privacy and system integrity. In highly sensitive environments such as ICUs, this setting introduces a significant security vulnerability and should be avoided in production systems. In contrast, when the setting is set to `False`, the system enforces strict participant admission control. Only nodes presenting valid, trusted identity credentials during the discovery handshake are allowed to join the DDS domain. As a result, any unauthorized participant attempting to connect will be silently rejected. This configuration strengthens the system's overall security baseline. Table 1 summarizes the expected system behavior and security implications associated with `allow_unauthenticated_participants`.

Table 1. Effects of the `allow_unauthenticated_participants` Setting.

Source: [18]

Value	System Behavior	Security Impact
True	Allows unauthenticated participants to join the DDS domain.	Increases security risks due to potential unauthorized access.
False	Requires authentication for all participants before joining.	Enhances security by ensuring only verified participants are allowed.

Once a participant has been authenticated, the next stage involves determining whether the system performs additional authorization checks, as governed by the `enable_join_access_control` setting. It is defined in the permission file and governs whether the system performs an additional layer of authorization after a participant successfully authenticates. It ensures that only participants explicitly permitted by the governance configuration are allowed to join the DDS domain. In the context of the ICU MetaCares Monitor Station, enabling this setting is essential for implementing fine-grained access control. For instance, even if a device or user presents valid credentials, it should not automatically be permitted to participate in the DDS domain unless explicitly authorized.

With `enable_join_access_control` set to `True`, the system consults the allow/deny rules defined in the governance file to assess the participant's eligibility to join. This ensures that only authorized entities are permitted to determine whether the participant is permitted to join the domain. This prevents unauthorized but technically valid nodes, such as non-clinical devices, from being included in the real-time data distribution infrastructure. Conversely, if the setting is disabled, all authenticated participants are allowed to join the DDS domain without additional authorization checks. Table 2 summarizes the expected system behavior and security implications associated with `enable_join_access_control`.

Table 2. Effects of the `enable_join_access_control` Setting.

Source: [18]

Value	System Behavior	Security Impact
True	Requires authentication and authorization; rejects invalid nodes.	Enhances domain-level access control and reduces attack surface.
False	All participants can join the DDS domain without access control enforcement.	Increases risk by exposure if topic-level permissions are not strictly configured.

Together, these two plugin rules establish a multi-layered defense mechanism for DDS domain admission. The former governs whether identity credentials are required during the initial discovery process, while the latter introduces an additional layer of authorization based on explicit permissions. The interplay between these two rules creates the initial gatekeeping structure that determines whether a participant may be admitted to the DDS domain, laying the groundwork for subsequent topic-level control.

Scenario 2: Sensitive Medical Data Exposed Due to Missing Read Access Control

In the simulated ICU system, patient data is constantly transmitted between bedside devices, nurse stations, and monitoring systems. If a maintenance laptop temporarily connected to the network unintentionally receives patient medical records or waveform data, it may result in a breach of patient confidentiality and potentially also lead to violations of

data protection regulations. Although authentication ensures that only verified devices and users can join the domain, it does not guarantee that every participant is permitted to access all data. The corresponding scenario is illustrated in Figure 5.

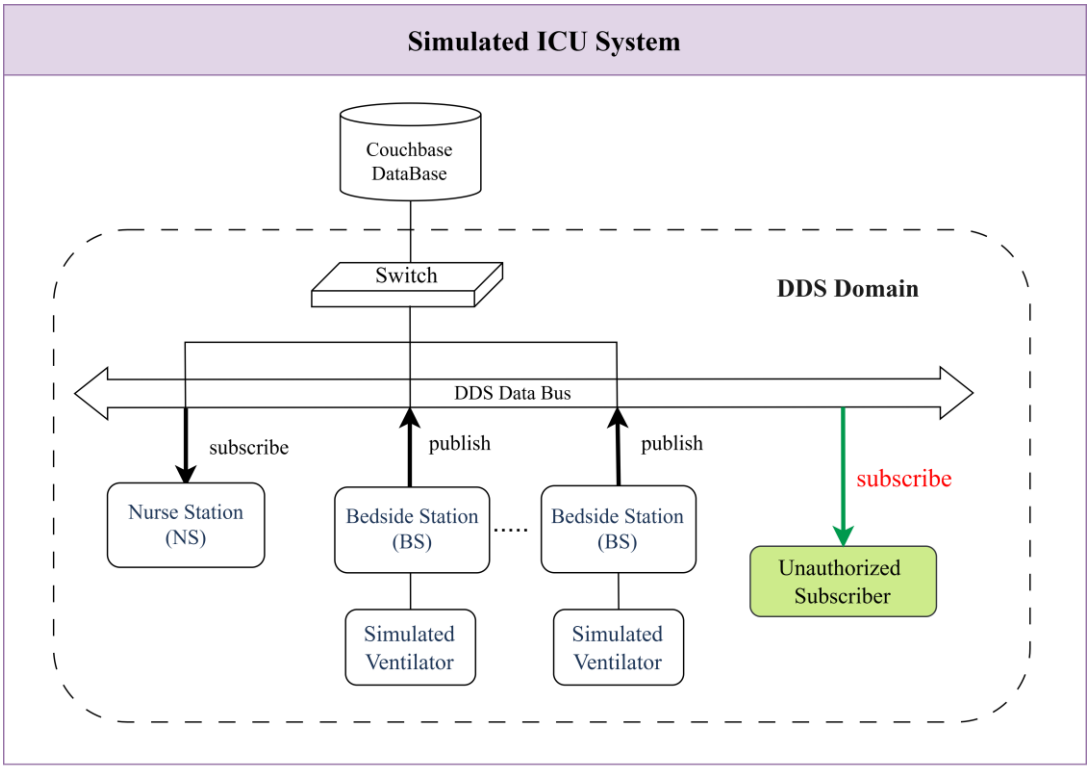
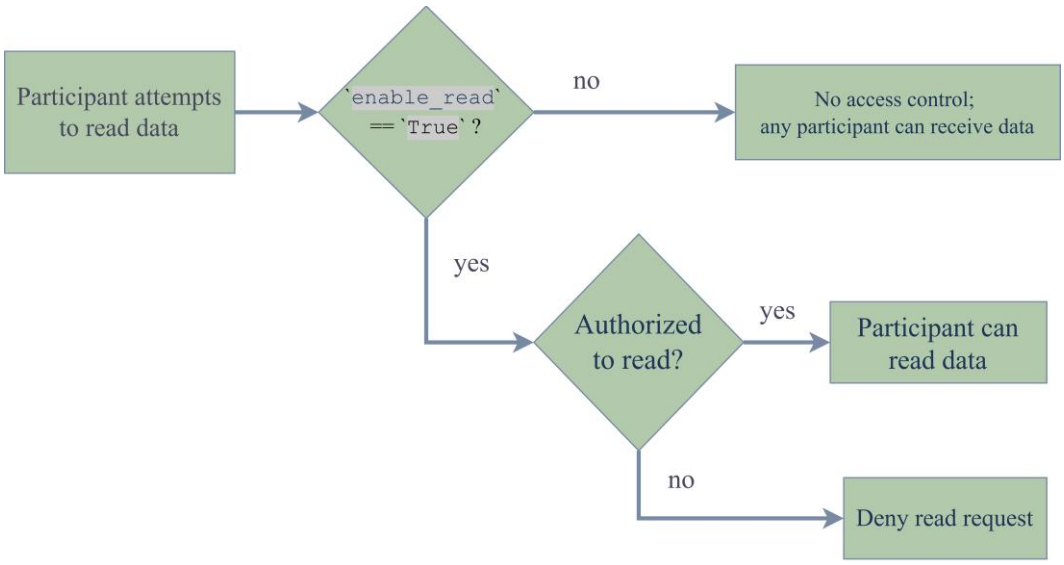


Figure 5. Unauthorized data access via non-clinical device connection.

A setting in Access Control Plugin called `enable_read_access_control` can prevent this type of data exposure. It governs whether the system enforces authorization checks before allowing a participant to subscribe to a DDS topic. This ensures that sensitive data remains accessible only to authorized entities, reinforcing the principle of least privilege within the system. The system can activate the `enable_read_access_control` setting within the Access Control Plugin. Figure 6 illustrates the protection mechanism enforced by this setting, showing how unauthorized subscription attempts are blocked.



- `enable_read` == `enable_read_access_control`

Figure 6. Participant read access control flow with security plugin.

When `enable_read_access_control` is set to `True`, the DDS middleware enforces the subscription rules defined in the permissions file, allowing only explicitly authorized participants to subscribe to specific topics. This ensures that confidential patient data is only disseminated to intended recipients, such as nurse stations or clinical decision-support modules. In contrast, if this setting is disabled, any authenticated participant can subscribe to all topics within the DDS domain, regardless of their role or intended use of the data. This opens the system to significant privacy and operational risks. Unauthorized subscribers, whether introduced through misconfiguration or through deliberate intrusion, may be able to passively monitor sensitive patient data. This violates the principles of medical confidentiality and increases the risk of data leakage or unauthorized analytical use. For instance, if an unapproved client receives real-time respiratory data from ventilators, it may lead to unintended secondary use or even facilitate broader data exfiltration. These scenarios compromise data protection requirements established by healthcare information governance policies and undermine the overall trust in the system's confidentiality and integrity. Therefore, enabling read access control is essential for enforcing data minimization principles, ensuring privacy, and maintaining regulatory compliance in real-time medical environments.

The implications of enabling or disabling `enable_read_access_control` are summarized in Table 3.

Table 3. Effects of the `enable_read_access_control` setting.

Source: [18]

Value	System Behavior	Security Impact
True	Read access is enforced; participants must be authorized to read protected topics.	Prevents data leakage by blocking unauthorized reads.
False	All participants can read topics regardless of permissions.	Increases risk of exposing sensitive data to unauthorized entities.

Scenario 3: Risk of Harmful Data Injection Without Writing Control

In a typical ICU setting, various systems, including ventilators, patient monitors, and infusion pumps, continuously publish critical data. These data are consumed by monitoring systems, electronic medical records (EMR), and decision-support tools. If a misconfigured device or unauthorized application gains the ability to publish to these topics, it may inject incorrect or falsified data into the system. For instance, an unverified maintenance terminal accidentally configured as a publisher could overwrite vital patient metrics such as heart rate or oxygen levels. In high-stakes environments such as ICUs, such breaches compromise the integrity of clinical workflows and endanger patient outcomes. The corresponding scenario is illustrated in Figure 7.

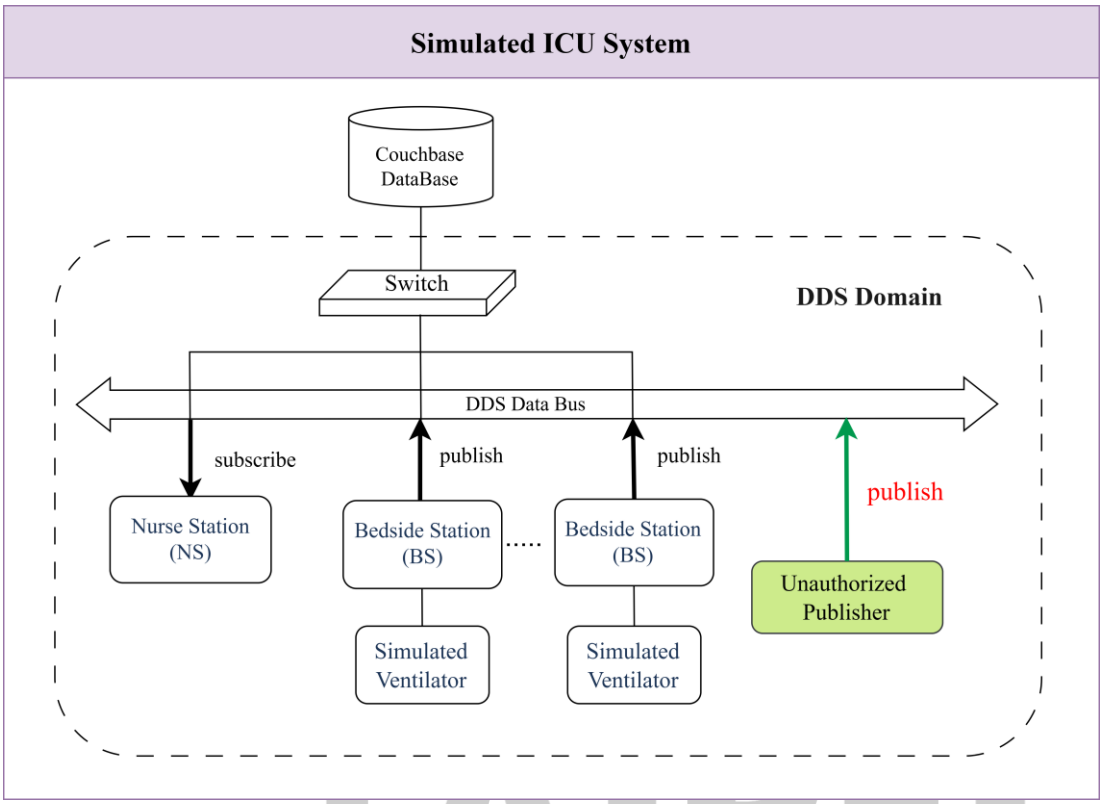
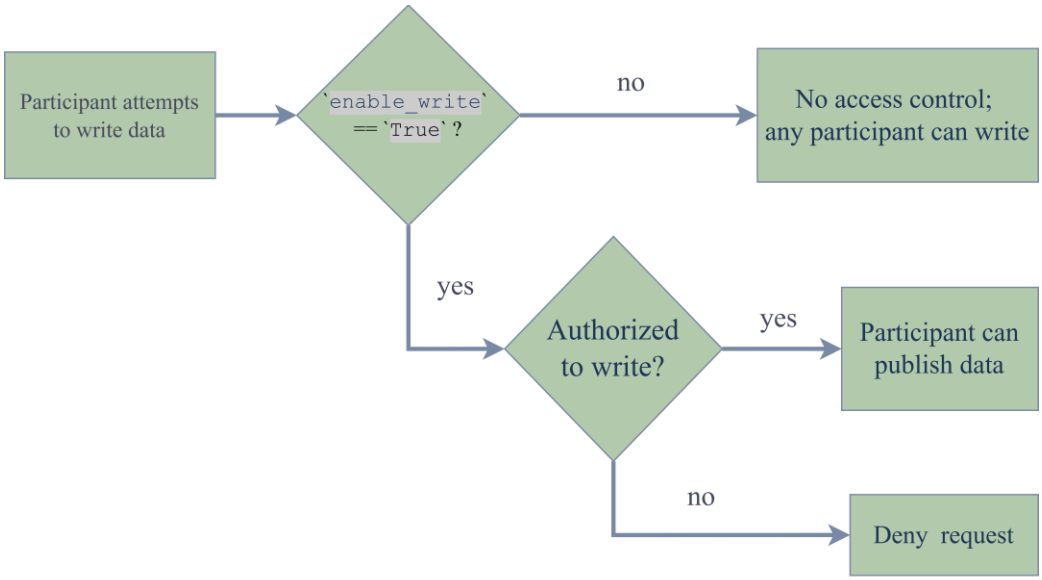


Figure 7. Sensitive medical data exposed without read access control.

Such risks are especially concerning in DDS-based communication environments where multiple nodes interact in real time. Authentication mechanisms can ensure that only known participants are allowed into the domain, but they do not restrict what actions each participant is allowed to perform. To address this, the Access Control Plugin includes a setting called `enable_write_access_control`, which determines whether the system enforces authorization checks before permitting a participant to publish data to a DDS topic. This mechanism ensures that only explicitly authorized participants can write to sensitive topics, preserving both the accuracy and trustworthiness of the system's data. Figure 8 illustrates the protection mechanism enforced by this setting, showing how unauthorized publication attempts are blocked.



- `enable_write` == `enable_write_access_control`

Figure 8. Participant's write access control flow under the security plugin.

When `enable_write_access_control` is `True`, it enforces strict adherence to the publishing permissions defined in the DDS governance configuration, thereby preventing unauthorized entities from injecting unverified or potentially malicious data into critical communication channels. Conversely, if this setting is `False`, any authenticated participant may publish to any topic within the DDS domain regardless of their role or operational relevance. In such cases, the system becomes vulnerable to misinformation, whether due to misconfiguration or intentional interference. For example, the injection of falsified alarm data or spoofed vital signs could lead to incorrect clinical responses, trigger unnecessary interventions, or obscure genuine emergencies. Therefore, enabling write access control is a fundamental requirement for secure, dependable, and regulation-compliant operation in real-time medical systems. The implications of enabling or disabling `enable_write_access_control` are summarized in Table 4.

Table 4. Effects of the `enable_write_access_control` Setting.

Source: [18]

Value	System Behavior	Security Impact
True	Enforces write permissions; only authorized participants can publish.	Protects integrity by blocking unauthorized or malicious writes.
False	Without writing checks, any authenticated participant can publish.	Allows unauthorized data injection, risking integrity and system stability.

Scenario 4: Sensitive Data Exposed Due to Insufficient Encryption Settings

In the DDS architecture, the Real-Time Publish-Subscribe (RTPS) protocol underlies the communication between participants. It defines the structure and delivery of messages, including data payloads and associated metadata, across the DDS domain. Since RTPS messages are transmitted over the network in real-time and can contain sensitive medical information, such as patient vitals or alarms, securing these messages is essential to prevent passive interception or data tampering. Even with authentication and access control in place, the absence of encryption in real-time patient data flows may still expose DDS RTPS messages to passive interception. In practice, this can occur if an outdated monitoring device or a diagnostic workstation with background logging may inadvertently capture unencrypted transmissions, such as patient vitals or alarm notifications. As illustrated in Figure 9, authenticated participants may still exchange plaintext messages, leaving sensitive information vulnerable to eavesdropping and data breaches.

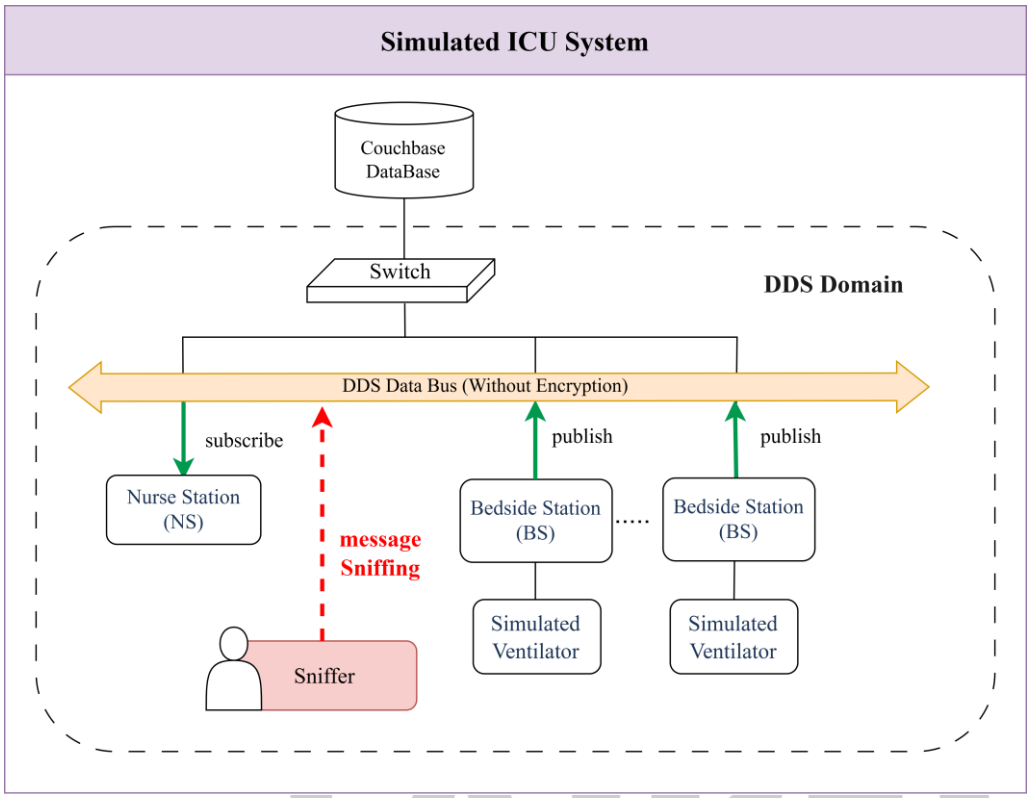


Figure 9. Patient data exposed due to lack of cryptographic protection.

To mitigate such risks, RTI Security Plugins provide several protection settings that determine how different parts of the DDS communication are secured. Among them, `rtps_protection_kind`, which controls whether and how the entire RTPS message is encrypted. This setting offers multiple protection levels to allow users to choose the appropriate level of security based on specific operational needs, like `NONE`, `SIGN`, `ENCRYPT`.

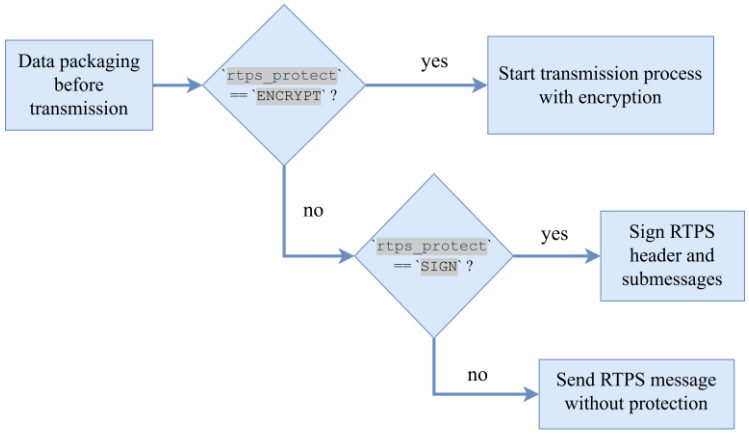
The `data_protection_kind` specifically governs the encryption and integrity protection of the data payload itself. Setting `data_protection_kind` to `ENCRYPT` ensures that the actual content of the messages exchanged between participants is secured, preventing unauthorized interception and reading. This is critical in environments like ICU systems, where patient data confidentiality and integrity are paramount.

In this scenario, setting `rtps_protection_kind` to `ENCRYPT` ensures that entire RTPS message, including headers and payload, is protected from passive interception from

passive interception, making it unreadable to unauthorized listeners even if they gain access to the network. Furthermore, setting `data_protection_kind` to `ENCRYPT` specifically secures the actual data payload itself, providing an additional layer of protection against unauthorized access during transmission. This layered encryption is essential in the ICU system, where the integrity and confidentiality of real-time patient data are critical to clinical safety.

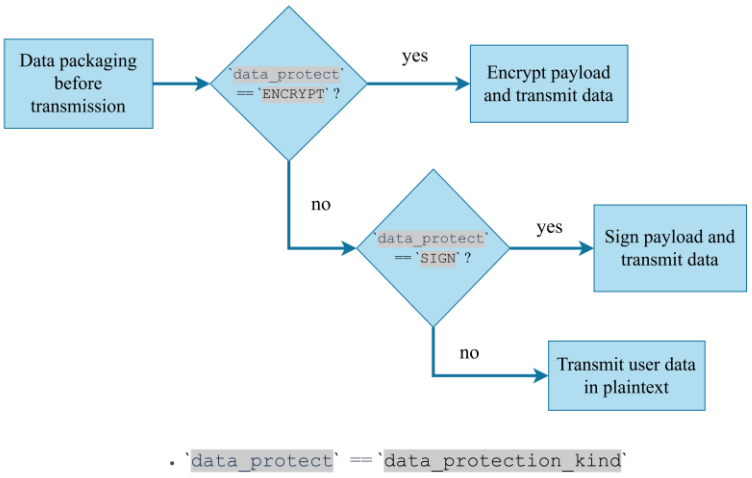
In addition to encrypting the message content, DDS also allows protection of the accompanying metadata using the `metadata_protection_kind` rule. Metadata includes topic names, QoS configurations, and routing information. If left unprotected, these details could expose the structure of the system and aid malicious actors in crafting targeted attacks. By enabling metadata encryption alongside RTPS message protection, the system achieves a more comprehensive defense against both content leakage and traffic analysis. These settings together establish layered confidentiality within the DDS domain. Each setting contributes to a comprehensive defense against different types of attacks. Misconfigurations or underestimating the importance of these settings may leave critical systems vulnerable despite authentication and access controls being in place. Figure 10. illustrates the conditional handling flow under different protection kinds. Each subfigure presents the logic flow for whether encryption, signing, or no protection is applied to RTPS messages, user data, and metadata, respectively, before transmission.

a) RTPS protection;



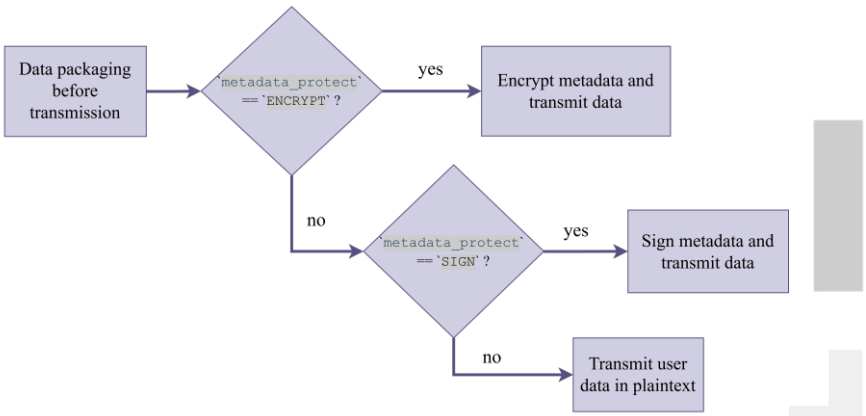
- `rtps_protect` == `rtps_protection_kind`

b) Payload protection;



- `data_protect` == `data_protection_kind`

c) Metadata protection;



- `metadata_protect` == `metadata_protection_kind`

Figure 10. Protection procedure for data transmission under different protection kinds: (a) RTPS protection; (b) User data protection; (c) Metadata protection.

3.2 Experimentation

This section describes the experimental environment established to evaluate the impact of different DDS Security configurations on data transmission performance in a simulated ICU scenario. Section 3.2.1 introduces the hardware and software components required to build the test environment. Section 3.2.2 describes the setup process. Section 3.2.3 explains the implementation of the test cases. Together, these elements support a structured investigation of performance under realistic conditions.

3.2.1 Hardware/Software Requirements

The simulated environment is configured with specific hardware and software resources to ensure the proper deployment of the experimental framework and to maintain the consistency and reliability of testing procedures. A stable local network is established to support reliable communication between bedside stations and the nursing station. Software components, including the middleware platform and monitoring tools, are carefully selected to support secure data exchange, performance observation, and systematic experimentation. The detailed hardware and software requirements are summarized in Table 5 to 7, respectively. Table 5 summarizes the hardware used in the setup. Table 6 and 7 outline the software stack supporting the experiment, including the operating system, middleware, data management platforms, and additional monitoring tools.

Table 5. Hardware requirements for the experimental setup.

Component	Description	Specification	Role in Experiment
Switch	Facilitates network communication between devices.	TP-link 8-port Gigabit Desktop Switch TL-SG108	Connects BSs and the Ns within the network.
DB	Serving as database node.	Intel i5-12500H, 32GB RAM	Hosts Couchbase Server and Sync Gateway for data storage and sync.
NS	NUC device serving as the nurse station.	Intel NUC11PAH (Intel i5-1340P, 16G RAM)	Receive and monitor patient data from BSs.
BS-1	NUC device serving as the ICU bedside station 1.	Intel NUC11PAH (Intel i5-1315U, 8G RAM)	Receives simulated ventilator data and forwards it to the NS.
BS-2	NUC device serving as the ICU bedside station 2.	Intel NUC11PAH (Intel i5-1340p, 8G RAM)	Receives simulated ventilator data and forwards it to the NS.

Table 6. Third-party open-source software.

Component	Description	Version	License	Role in Experiment
Ubuntu [27]	Operating system.	20.04 LTS	GPL-2+, GPL-3+	Provides the OS environment for all experimental devices.
OpenSSL [28]	Provides TLS/SSL and authentication support.	3.0.10	Apache License 2.0	Serves as the underlying cryptographic engine for RTI Connext Secure.
Qt Library [29]	C++ application framework.	6.2.4	LGPL	Supports development of MetaCares Monitoring Station user interface.
QWT [30]	GUI components for data visualization based on Qt.	6.2.0	LGPL	Used for plotting real-time curves in the GUI.
Couchbase Server [31]	Distributed NoSQL database.	7.0.2	BSL1.1	Stores transmitted physiological data.
Couchbase Sync Gateway [33]	Sync layer for distributed Couchbase data.	3.0.3	Apache License 2.0	Enables real-time and offline data synchronization.
Couchbase Lite [32]	Embedded NoSQL database for edge devices.	3.1.9	Apache License 2.0	Supports offline operation and secure sync via Sync Gateway.
Wireshark [34]	Network protocol analyzer for packet inspection.	3.2.3	GPL v2	Used to inspect DDS traffic, encryption, and timing.

Table 7. Software requirements for the experimental setup.

Component	Description	Version	License	Role in Experiment
RTI Connext Secure	DDS middleware with built-in security.	6.1.2	Commercial	Handles secure data exchange between BS and NS.
MetaCares Monitoring Station	Physiological data monitoring and relay system.	Preview-1.1.0	Commercial	Gathers patient data and distributes it through DDS for further processing.
Restful API	Web communication interface.	2.5.3	Commercial	Facilitates system integration and remote communication.
RTI Admin Console	Graphical tool for inspecting DDS communication.	6.1.2	Commercial	Monitors DDS activities and assists in participant configuration.
Simulated Ventilator	Application that publishes patient data via DDS.	Internal	Proprietary	Emulates a real ventilator to generate continuous ICU data flow
Simulated External Subscriber	Test application for simulating a subscriber.	Custom	Proprietary	Used to simulate subscriber behavior for testing security scenarios.
Simulated External Publisher	Test application for simulating a publisher.	Custom	Proprietary	Used to simulate publisher behavior for testing security scenario.

3.2.2 Experiment Setup

Following the specification of hardware and software requirements in Section 3.2.1, the installation and configuration procedures steps for the experiment are described in this section. A structured overview of the setup follows.

A. Installation of MetaCares Monitoring Station and Supporting Services:

The MetaCares Monitoring Station and its associated services were installed on an Intel NUC device running Ubuntu 22.04. The setup includes Couchbase Server and Couchbase Lite Sync Gateway, which support local data storage and synchronization. The following steps outline the installation process.

- MetaCares Monitoring Station

  MetaCares Monitoring Station is a commercial software product. It is deployed using an installation package provided by the vendor, along with a corresponding setup guide. As these resources are not publicly available, the installation is carried out according to the procedures specified in the vendor's documentation.

- Couchbase Server, Couchbase Lite and Couchbase Sync Gateway

  Couchbase Server and Sync Gateway are installed on the DB to provide centralized data management and synchronization services. Couchbase Lite and Sync Gateway clients are installed on each NUC device to locally cache data and enable real-time synchronization with the central server.

  Before installing individual components, the Couchbase repository must be configured:

  1. Open the terminal and install curl:

     ```
     $sudo apt install curl
     ```

  2. Download the Couchbase release package:

     ```
     $curl -O https://packages.couchbase.com/releases/couchbase-release/couchbase-release-1.0-noarch.deb
     ```

  3. Install the downloaded release package:

     ```
     $sudo apt install ./couchbase-release-1.0-noarch.deb
     ```

  4. Updating the package list:

```
$sudo apt update
```

On DB device:

1. Install Couchbase Server with version 7.0.2:

```
$sudo apt install couchbase-server-community=7.0.2-ubuntu20.
04
```

2. Download the Couchbase Sync Gateway package:

```
$curl -O https://packages.couchbase.com/releases/couchbase-s
ync-gateway/3.0.0/couchbase-sync-gateway-community_3.0.0_x86
_64.deb
```

3. Install the Couchbase Sync Gateway package:

```
$sudo apt install ./couchbase-sync-gateway-community_3.
0.0_x86_64.deb
```

On the NUC device:

1. Install the Couchbase Lite development package:

```
$sudo apt install libcblite-dev-community
```

2. Download the Couchbase Sync Gateway package:

```
$curl -O https://packages.couchbase.com/releases/couchbase-s
ync-gateway/3.0.0/couchbase-sync-gateway-community_3.0.0_x86
_64.deb
```

3. Install the Couchbase Sync Gateway package:

```
$sudo apt install ./couchbase-sync-gateway-community_3.
0.0_x86_64.deb
```

- Qt Library and QWT

1. Install the Qt development libraries:

```
$sudo apt install qt5-default libqt5x11extras5-dev
```

2. install the QWT library:

```
$sudo apt install libqwt-qt5-dev
```

B. Setup of RTI Connext Secure and Dependencies

- RTI Connext Secure

RTI Connext Secure is a commercial product that provides security features for the

DDS protocol. After obtaining the necessary license, the installation process follows the official RTI Security Plugins Installation Guide [35], which includes setting up OpenSSL and integrating security plugins. The installation also involves configuring relevant environment variables to ensure that the security plugins function correctly within the RTI Connext framework.

C. Installation of Monitoring and Analysis Tools

- RTI Admin Console

  RTI Admin Console is included as part of the RTI Connext installation package. It provides a graphical interface for visualizing and diagnosing the behavior of DDS systems, including the discovery of participants, data writers, and data readers. This tool is primarily used during the experiment to verify that topics are being correctly published and subscribed to, and to monitor real-time communication between nodes.

- Wireshark

  Wireshark is a widely-used network protocol analyzer capable of capturing and displaying packets in real time. In this experiment, Wireshark is employed to examine DDS communication at the network level, particularly to observe the structure of secure data transmissions when using RTI Connext Secure. It also serves as a verification tool to confirm encryption behavior and evaluate packet-level transmission characteristics. It is installed using the standard package manager:

  1. Open the terminal and update the package list:

     ```
     $sudo apt update && sudo apt upgrade
     ```

  2. install Wireshark:

     ```
     $sudo apt install wireshark
     ```

3.2.3 Implementation

To evaluate the effectiveness of security plugin policies under various scenarios, we developed and deployed multiple custom DDS applications within a simulated ICU system. These applications include DDS publishers and subscribers that simulate unauthorized or access-restricted attackers and are executed with specific security plugin settings applied at

the system level. By simulating unauthorized or restricted access attempts, they help verify whether data transmission and access are effectively protected according to the defined security requirements.

Every security plugin rule applied in each test scenario is detailed in Section 3.1.2, and the expected behavior for transmission and access is determined by the risks and corresponding protection goals defined for each scenario. Each scenario was executed in two configurations: one with the security plugin disabled to serve as a baseline, and the other with the designated plugin rules enabled. This setup allowed us to directly compare the system's behavior and performance with and without security enforcement. To ensure meaningful and representative experimental results, the security plugin configurations for each scenario were carefully selected to activate relevant rules corresponding to the specific threat being simulated. This approach avoids trivial or ineffective configurations such as enabling multiple rules simultaneously with conflicting settings, which would effectively replicate the disabled baseline scenario. By applying the most pertinent configuration per scenario, the comparison between enabled and disabled security plugin states clearly reflects the impact of security enforcement.

In the simulated ICU system, multiple DDS topics are used to transmit patient data, such as vital signs, ventilator parameters, and alarm messages. For testing purposes, we select one of the primary topics that continuously broadcasts real-time physiological information, referred to as topic A. This topic is chosen because it reflects typical patient data flows in an ICU and is updated frequently during normal operation. Therefore, it serves as a representative and sensitive data stream to evaluate how well different security configurations can protect access to critical patient information.

During the experiments, besides verifying the effectiveness of the security plugins in protecting the system, we also used RTI Admin Console, Wireshark, and System Monitor to closely observe communication status and packet behavior. We recorded system resource usage, including CPU utilization, memory consumption, and data transmission latency as performance indicators, to comprehensively assess the impact of security plugins on system performance. The DDS Domain ID used in the experiments was set to X, and served as the basis for subsequent configuration and observation. Additionally, appropriate security configuration files and identity credentials were applied according to scenario requirements to support thorough evaluation and discussion of the results in later case studies. Below are descriptions of the steps for Scenario 1 to 4.

Scenario 1: An unauthorized participant joins the communication domain

Case 1: Security Plugin Disabled

When the system is configured without the security plugin**,** we create an unauthorized subscriber and provide it with a permissions file that only allows access to Domain Y. We then use this participant to attempt to access Domain X, which is designated for the simulated ICU system, and observe whether it could receive patient-related data. The subscriber attempts to subscribe to topic A. In DDS, successful subscription to a topic implies that the participant has completed the discovery process and joined the communication domain. Under these conditions, we monitor whether the participant can receive data. Since no security enforcement is enabled, the participant can join Domain X and successfully subscribe to patient-related topics.

Case 2: Security Plugin Enabled

In this case, the system is configured with the security plugin enabled. Specifically, both `allow_unauthenticated_participants` is set to `False` and `enable_join_access_control` is set to `True`, forming a strict security setting that denies any participant lacking proper authentication or authorization. Although there are multiple possible combinations of these two settings, this study focuses on the most restrictive configuration commonly adopted in practice, especially in medical environments where strong protection of sensitive data is essential.

We create a subscriber using the same permission file as in Case 1, which only grants access to Domain Y. However, unlike the previous case, the system's governance file specifies that only authenticated participants are allowed and that explicit permission is required to join a domain. Under this configuration, the subscriber attempts to access Domain X, specifically by trying to subscribe to topic A. Because its permissions do not include Domain X, it fails during the authentication and authorization stages. As a result, it cannot join the domain or receive any patient-related data. This strict configuration will serve as the security baseline for subsequent scenarios.

Scenario 2: Sensitive Medical Data Exposed Due to Missing Read Access Control

Case 3: Security Plugin Disabled

When the system is configured without the security plugin, any participant that joins the communication domain can subscribe to available topics without restriction. In this case, we create an unauthorized subscriber that lacks read access permissions and allow it to connect to the system. The subscriber attempts to subscribe to topic A. Since read access control is not enforced, the subscriber successfully receives the data.

Case 4: Security Plugin Enabled

In this case, the system is configured with the security plugin enabled, and read access control is enforced. The governance file includes `enable_read_access_control` set to `True`, and the permissions file specifies which topics each participant is allowed to read. An unauthorized subscriber, using the same permission file as in Case 3, attempts to subscribe to topic A, which carries sensitive physiological data. Because the subscriber does not have read permission for this topic, the security plugin blocks the data delivery. As a result, the participant is unable to receive any patient-related data. This confirms that enabling read access control effectively prevents unauthorized access to sensitive information.

Scenario 3: Risk of Harmful Data Injection Without Writing Control

Case 5: Security Plugin Disabled

With the security plugin disabled, any participant can publish to any topic without validation. We simulate an unauthorized publisher that sends data to Topic A, which is used by the simulated ICU system to transmit patient physiological data. Since write access control is not enforced, the system accepts and disseminates the data to subscribers. We confirmed the successful injection of unauthorized data by observing the published messages appearing in the RTI Admin Console.

Case 6: Security Plugin Enabled

In this case, the system is configured with the security plugin enabled, and write access control is enforced. The governance file includes `enable_write_access_control` set to `True`, and the permissions file specifies which participants are authorized to publish to specific topics. We create an unauthorized publisher without permission to write to Topic A, the same topic used in Case 5 for transmitting patient physiological data. When this publisher attempts to publish data, the security plugin blocks the operation based on the configured access control rules. We verify that no unauthorized messages appear in the system by monitoring Topic A through RTI Admin Console, confirming that write access control effectively prevents harmful data injection.

Scenario 4: Sensitive Data Exposed Due to Insufficient Encryption Settings

Case 7: Security Plugin Disabled

When the security plugin is not enabled, data transmitted across the network is not encrypted by default. In this scenario, we simulate normal communication within the system and use Wireshark to monitor the network traffic. Specifically, we capture packets containing updates to Topic A, which carries patient physiological information. Upon inspection, we find that the data content is visible in plaintext.

Case 8: Security Plugin Enabled

In this case, the security plugin is enabled with encryption configured. The governance file specifies `rtps_protection_kind` and `data_protection_kind` are both set to `ENCRYPT`, and `metadata_protection_kind` is also configured as `ENCRYPT` to secure the actual data payload. The permissions file includes the necessary cryptographic tokens. Using Wireshark, we monitor network traffic for Topic A, which carries patient physiological data. The captured packets show that the data is transmitted in encrypted form, with no readable plaintext visible.

With the experimental implementation described, the following chapter will provide a detailed analysis of the results, demonstrating the effectiveness and performance impact of the security plugins.

Chapter 4 Result

This section presents the experimental results based on four carefully designed security threat scenarios. Each scenario simulates a specific risk that may arise during medical data transmission and investigates the differences in system performance and data protection when the Security Plugin is enabled or disabled. By evaluating settings such as authentication, access control, and encryption, the analysis demonstrates how these mechanisms reduce threat risks and contribute to maintaining system stability. These results offer insights into the necessity and impact of incorporating security mechanisms in real-time medical communication environments.

In each scenario, the results from both cases are first presented. The effectiveness of threat mitigation is considered to determine whether the intended security goals are achieved. A comparison is then conducted to evaluate the additional system overhead introduced by the plugin. Specifically, CPU usage is used to assess processing load, while memory consumption reflects the impact on system memory. Latency and throughput are also analyzed to assess the performance of data transmission.

Scenario 1: An unauthorized participant joins the communication domain

The results of Case 1 and Case 2 are discussed as detailed below:

In Case 1, where the security plugin is disabled, the system does not enforce authentication or domain access verification. Even when the unauthorized subscriber lacks valid credentials for domain X, the system does not verify whether the subscriber has permission to join domain X. From the result that the unauthorized subscriber successfully receives patient data, the outcomes shows that it has effectively joined Domain X. This indicates that, in the absence of identity verification and permission evaluation, the system cannot prevent untrusted entities from accessing sensitive communication channels.

In contrast, in Case 2, where a simulated ICU system enabled the Security Plugin rules, it enforces authentication and domain access control policies. During the participant discovery phase, the system checks the configuration rule in the governance file and permissions file to verify whether the joining participant has permission to access Domain X. Since the unauthorized participant lacks the required domain credentials in its signed files, the subscription attempt fails and returns an error message such as "*participant not allowed: no*

rule found for the participant's domainId; default DENY". This error confirms that the system rejects the connection attempt and prevents the subscription from proceeding, effectively mitigating the type of risk described in Scenario 1.

In addition to confirming the security effect, we also observe the resource usage on both the BS and the NS through the System Monitor and RTI Admin Console. Our focus is on evaluating processor load and process-level load distribution while the system operates under secure and non-secure configurations. Although two BSs are deployed during the experiment, their hardware and software environments were identical. Observed system behavior, including CPU usage and process activity, showed consistent patterns across both units. Therefore, the performance results presented here are based on one representative bedside system. The observed results are summarized in Table 8, which compares major resource usage metrics under both configurations. Notably, packet loss remained at 0% in both cases, indicating that enabling authentication did not affect data transmission reliability.

Table 8. Comparison of system performance with and without the plugin in Scenario 1.

	NS		BS	
Metric	Case 1	Case 2	Case 1	Case 2
CPU (%)	6	6	12	12
Memory (Mib)	49.9	58.7	59.1	66.0
Latency (ms)	36.85	130.85	36.85	130.85
Throughput (bytes/s)	5909.2	5909.2	5909.2	5909.2

The table shows the system performance under secure and non-secure configurations in Scenario 1. CPU usage remained consistent across both conditions, with negligible differences observed on both the NS (6%) and BS (12%). Memory consumption increases 8.8 Mib on the NS and 6.9 Mib on the BS. Given that the NS is equipped with 16 GB of RAM and the BS with 8 GB, the increases represent only 0.05% and 0.08% of total memory, respectively. This indicates that enabling the plugins does not impose a significant computational burden. Such overhead is negligible and well within the system's capacity for real-time operations. A noticeable difference was observed in latency, increasing from 36.85 ms to 130.85 ms after enabling authentication. This increase is primarily attributed to the

additional verification steps during the domain join process. However, the latency overhead occurs only during the initial handshake phase and does not persist during ongoing data transmission. Importantly, throughput remained unchanged at approximately 5909.2 bytes/s (equivalent to 6.8 samples/s), indicating that data delivery performance was not compromised. These results suggest that while security mechanisms introduce minor overhead during initialization, they do not affect sustained system throughput or resource efficiency, making them suitable for secure, real-time medical communication.

Scenario 2: Sensitive Medical Data Exposed Due to Missing Read Access Control

The results of Case 3 and Case 4 are discussed as detailed below:

In Case 3, where read access control is not enabled, the system does not enforce restrictions on who can subscribe to Topic A, which contains sensitive medical content. Although the unauthorized subscriber lacks valid permissions, it is still able to successfully subscribe to Topic A and receive patient-related data. This finding indicates that the absence of read access control causes the system to fail in verifying subscriber permissions and allows unverified recipients to access protected information. As a result, sensitive data is exposed, highlighting a critical vulnerability in configurations lacking subscriber-level authorization.

With the read access control rule enabled in Case 4, the system checks permissions before delivering the data and restricts topic access based on predefined policies. Since the participant who attempts to subscribe with the permission in Topic A, the system evaluates the permissions file during the subscription phase and rejects the data delivery accordingly. The system returns an error message stating: "*endpoint not allowed: no rule found; default DENY.*" As a result, the unauthorized subscriber does not receive any content from Topic A. This confirms that the read access control mechanism effectively enforces topic-level restrictions, preventing data exposure to unverified recipients and preserving the confidentiality of sensitive medical information.

This confirms the importance of enforcing read-level access policies in healthcare systems. Additionally, we observe any performance overhead introduced by access control enforcement. These metrics are summarized in Table 9, which compares system performance under secure and non-secure conditions for Scenario 2. As in Scenario 1, packet loss remains at 0% in both configurations, and resource usage is monitored on a representative BS due to identical hardware/software setups across BS units.

Table 9. Comparison of system performance with and without the plugin in Scenario 2.

	NS		BS	
Metric	Case 3	Case 4	Case 3	Case 4
CPU (%)	6	6	12	12
Memory (Mib)	50.7	58.6	58.2	65.9
Latency (ms)	36.85	90.99	-	-
Throughput (bytes/s)	5909.2	5909.2	5909.2	5909.2

From Table 9, CPU usage remained consistent across both configurations, indicating that enabling read access control did not introduce notable processing overhead. Memory usage showed a slight increase, approximately 7.9 Mib at the NS and 7.7 Mib at the BS, after enabling the access control plugin. These increments represent approximately 0.05% and 0.09% of their total memory capacity, respectively. Latency increases from 54.1 ms following the activation of access control, primarily due to permission verification during the participant discovery and topic association phases. However, this added latency only occurs during the initial connection setup and does not affect ongoing data transmission. Such overhead is negligible and well within the system's capacity for real-time operations. This indicates that enabling access control does not impose a significant computational burden on the system. Throughput remained stable at approximately 5909.2 bytes per second under both configurations. Although the unauthorized subscriber in the secure configuration failed to receive any data, its DataReader did not successfully establish a connection with the protected topic. This had no effect on the publisher's transmission behavior, as throughput in the DDS architecture reflects data sent by the publisher rather than successful delivery to subscribers.

These findings demonstrate that enforcing read-level access policies in real-time healthcare systems can effectively enhance data protection without compromising system performance. The slight overhead introduced is justified by the prevention of unauthorized access to sensitive medical information.

Scenario 3: Risk of Harmful Data Injection Without Write Access Control

In Case 5, when write access control is disabled, the system does not verify whether a publishing participant holds the necessary permissions to write to Topic A. As a result, even

though the unauthorized publisher lacks write privileges, it successfully publishes data to Topic A. This behavior demonstrates a critical vulnerability, where harmful or misleading data can be injected into the system without validation, potentially compromising downstream processes such as automated alerts or clinical decision support.

In Case 6, with write access control enabled, the system enforces predefined policies that restrict publishing rights to authorized participants. When the unauthorized participant attempts to publish to Topic A, the system denies the request due to insufficient write permissions and returns the error message "*endpoint not allowed: no rule found; default DENY.*" The publishing attempt is blocked, thereby preserving the integrity of the communication flow and preventing the injection of unverified data.

A summary of system resource usage and latency differences between secure and non-secure configurations is provided in Table 10.

Table 10. Comparison of system performance with and without the plugin in Scenario 3.

	NS		BS	
Metric	Case 5	Case 6	Case 5	Case 6
CPU (%)	6	6	12	12
Memory (Mib)	50.7	59.0	58.1	65.8
Latency (ms)	-	-	36.85	228.79
Throughput (bytes/s)	5909.2	5909.2	5909.2	5909.2

CPU usage remained constant across both configurations, at 6% on the NS and 12% on the BS. And memory consumption increased slightly after enabling access control. Specifically, usage rose by 8.3 Mib on the NS and by 7.7 Mib on the BS. These increases represent approximately 0.05% of the NS's total memory and 0.09% of the BS's total memory. These results indicate that enabling the publishing permission mechanism did not introduce additional processing overhead. In this scenario, Latency showed a more noticeable increase, from 36.85 ms to 228.79 ms on both the NS and BS, representing an increase of nearly 192 ms. This increase is primarily attributed to the evaluation of publishing permissions during the data transmission attempt. However, as with previous scenarios, this added latency only affects the initial interaction phase and does not persist during normal data flow. For throughput, Despite the blocked publishing attempt by the unauthorized participant,

throughput remained unchanged at 5909.2 bytes/s, as the Publisher continued to operate and send data according to its configured rate. This confirms that the write access control mechanism affects only the authorization layer and not the core data production behavior of the system. Given that this mechanism effectively blocks unauthorized data injection, the minor performance overhead is a worthwhile trade-off for ensuring the integrity and trustworthiness of critical clinical data.

Scenario 4: Sensitive Communication Exposed Due to Insufficient Encryption Settings

In Case 7, the simulated ICU system operated without enabling encryption settings. Although the participant was authorized and could successfully join the communication domain and access Topic A, all transmitted data remained unprotected during transmission. Using tools such as Wireshark, we verified that patient-related content is visible in plaintext within captured network packets. This confirms that without encryption, sensitive medical information is vulnerable to interception by network-level attackers. The absence of encryption compromises data confidentiality and exposes patients to privacy risks. In Case 8, the system enabled encryption through the RTI Secure Plugin by configuring `rtps_protection_kind`, `data_protection_kind`, and `metadata_protection_kind`. This enforces secure communication at both the RTPS and topic levels. As a result, even though the participant retains access permissions, all transmitted content is protected. As shown in Figure 11, Packet inspection using Wireshark reveals the effect of this configuration: the data packets now include additional secure encapsulation layers such as `SRTPS_PREFIX`, `SEC_PREFIX`, and `SEC_POSTFIX`. The DATA payload becomes unreadable and appears in encrypted format, and the topic name is no longer identifiable. This indicates that the metadata has been signed and the payload encrypted, preventing unauthorized disclosure. All data exchanged between the publisher and subscriber is fully encrypted, rendering it inaccessible to unauthorized observers.

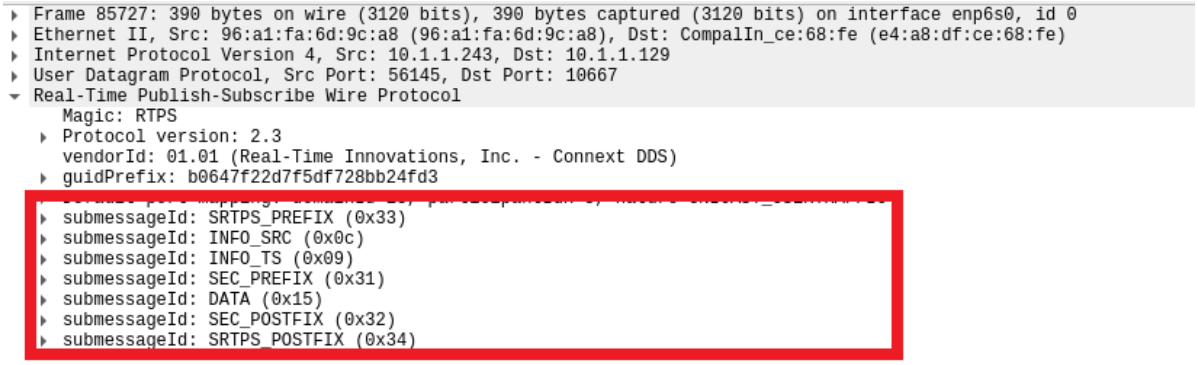


Figure 11. Secure DDS packet structure captured by Wireshark.

These findings confirm the successful application of encryption during transmission and demonstrate that enabling DDS-level encryption effectively mitigates the risk of data leakage, thereby enhancing the confidentiality of real-time healthcare communication.

Table 11 presents the system performance comparison under secure and non-secure configurations in Scenario 4, where encryption is applied.

Table 11. Comparison of system performance with and without the plugin in Scenario 4.

	NS		BS	
Metric	Case 7	Case 8	Case 7	Case 8
CPU (%)	6	6	12	12
Memory (Mib)	51.5	58.5	58.2	65.5
Latency (ms)	36.85	287.66	36.85	287.66
Throughput (bytes/s)	5909.2	5909.2	5909.2	5909.2

CPU usage remains unchanged across both settings, indicating that the additional cryptographic operations do not introduce a significant processing burden. Memory consumption increases by approximately 7.0 Mib at the NS and 7.3 Mib at the BS after enabling encryption, which represents less than 0.1% of their respective available capacities. These increments remain within acceptable operational thresholds. However, the latency shows a more noticeable increase, rising from 36.85 ms to 287.66 ms. This elevated latency

results from the computational overhead introduced by payload encryption and metadata signing, which are executed during the message processing pipeline. Despite the increased delay, this overhead only affects the initial stages of communication and does not impact the integrity or frequency of ongoing data exchange. Throughput remains stable at 5909.2 bytes/second, demonstrating that encryption does not degrade the system's ability to transmit data at the expected rate. The stability of throughput also reflects the fact that while subscribers receive encrypted data, the publisher continues to transmit at the configured rate, unaffected by whether the subscriber can decrypt or interpret the data. These results affirm that the use of DDS encryption mechanisms successfully strengthens confidentiality without compromising core system performance in a real-time medical communication environment.

Across all four scenarios, the results from Case 1 to Case 8 demonstrate that enabling Security Plugin configurations, specifically authentication, access control, and encryption, provides essential protection against a range of communication threats in real-time medical environments.

When the security plugins are disabled (Cases 1, 3, 5, and 7), the system allows unauthorized participants to join the domain, access sensitive patient data, publish potentially harmful messages, and transmit information over the network. These behaviors clearly expose the system to privacy breaches, data integrity risks, and unauthorized interference, which are unacceptable in critical healthcare applications. In contrast, enabling the relevant security mechanisms (Cases 2, 4, 6, and 8) successfully mitigates these risks. Authentication prevents unauthorized domain access, read and write access controls restrict topic-level permissions, and encryption ensures confidentiality of transmitted medical data. These protections operate effectively within the system and block unauthorized actions as expected. From a performance perspective, the activation of each plugin introduces only limited overhead. Authentication adds a one-time cost during participant discovery, access control incurs lightweight checks on publish or subscribe actions, and encryption moderately increases CPU load and latency. However, these effects remain within tolerable bounds and do not compromise the system's real-time communication capabilities.

Overall, the Security Plugin offers a balanced solution for enhancing communication security while maintaining system performance. For medical systems that transmit sensitive,

time-critical data, the implementation of these plugins is not only beneficial but necessary to ensure patient safety, data confidentiality, and system integrity.

Chapter 5 Conclusion & Future Work

This study investigated the impact of different security plugins on the performance and data protection of a DDS-based medical data transmission system. Through simulating common security threat scenarios in clinical healthcare environments, we verified that enabling security plugins such as authentication, access control, and data encryption effectively prevents unauthorized participant intrusion, avoids sensitive data leakage, and reduces the risk of malicious data injection. Meanwhile, we observed that these security measures, while ensuring data protection, have an associated overhead on transmission latency and resource consumption that is minimal, with increases generally within 0.1%. This suggests that DDS implements strong security in DDS-based medical systems without significantly compromising real-time performance.

The future work can focus on two main aspects. First, addressing more complex and targeted security threats beyond the basic attack scenarios examined in this study. Potential threats include insider attacks, forged governance or permission configurations, replay-based data injection, and timing-based attacks that exploit predictable system behavior. These threats often involve more subtle or context-aware methods that bypass conventional rule-based controls. Investigating how the Security Plugin mitigates these risks will help assess its effectiveness under more realistic and adversarial conditions. Second, extending Security Plugin support from single-domain to multi-domain environments. Securing communication across multiple DDS domains requires interoperable authentication, access control, and encryption mechanisms. This demands careful coordination of governance and permissions files, shared trust infrastructures, and robust key management strategies. Cross-domain identity validation, revocation handling, and time synchronization must also be considered to maintain secure and reliable data exchange. These directions not only strengthen practical requirements in hospital-wide deployments but also offer opportunities to enhance the DDS security framework for broader, large-scale applications.

References

[1] Object Management Group, *Data Distribution Service (DDS), version 1.4*, OMG Document No. formal/2015-04-10, 2015. [Online]. Available: https://www.omg.org/spec/DDS/1.4/.

[2] Z. Xiaowen, Z. Xiaogang, W. Shaoyuan and X. Ping, "Design and Implementation of Robot Middleware Service Integration Framework Based on DDS," in *Proc. 2022 IEEE Int. Conf. Real-time Comput. Robot. (RCAR)*, Guiyang, China, pp. 588-593. [Online]. Available: https://ieeexplore.ieee.org/document/9872212.

[3] Trend Micro and ADLINK, "A Security Analysis of the Data Distribution Service (DDS) Protocol," 2022. [Online]. Available: https://documents.trendmicro.com/assets/white_papers/wp-a-security-analysis-of-the-data-distribution-service-dds-protocol.pdf. [Accessed: Jul. 10, 2024].

[4] Maia Research, "Global Data Distribution Service (DDS) Market Research Report 2024-Competitive Analysis, Status and Outlook by Type, Downstream Industry, and Geography, Forecast to 2030," 2024. [Online]. Available: https://www.marketresearch.com/Maia-Research-v4212/Global-Data-Distribution-Service-DDS-37395378/. [Accessed: May. 23, 2025].

[5] ADLINK, "Using ADLINK DDS for Scalable, High Performance, Real-time Data Sharing Between Medical Devices in Next Generation Healthcare Systems," 2020. [Online]. Available: https://material.adlinktech.com/ADLinkFile/Publication/852/DDS_Medical_WP_FINAL.pdf. [Accessed: Oct. 12, 2024].

[6] Object Management Group, *DDS Security, version 1.2*, 2024. [Online]. Available: https://www.omg.org/spec/DDS-SECURITY/1.2/About-DDS-SECURITY.

[7] Robert Lemos, "Misconfigurations, Vulnerabilities Found in 95% of Applications," 2022. [Online]. Available: https://www.darkreading.com/application-security/misconfigurations-vulnerabilities-found-in-95-of-applications. [Accessed: Dec. 15, 2024].

[8] J. Ali, M. H. Zafar, C. Hewage, S. R. Hassan and R Asif, "The Advents of Ubiquitous Computing in the Development of Smart Cities—A Review on the Internet of Things (IoT)," *Electronics*, vol. 12, no. 4, pp. 1032, 2023. [Online]. Available: https://www.mdpi.com/2146884.

[9] A. Ioana and A. Korodi, "DDS and OPC UA Protocol Coexistence Solution in Real-Time and Industry 4.0 Context Using Non-Ideal Infrastructure," *Sensors*, vol. 21, no. 22, pp. 7760, 2021. [Online]. Available: https://www.mdpi.com/1369692.

[10] Object Management Group, *Data Distribution Service + Data Local Reconstruction Layer, version 1.4*, 2015. [Online]. Available: https://www.omg.org/spec/DDS-DLRL/1.4/About-DDS-DLRL.

[11] DDS Foundation, "What is DDS?" [Online]. Available: https://www.dds-foundation.org/what-is-dds-3/. [Accessed: Dec. 15, 2024].

[12] I. Stewart, A. Singh, and M. Toloui, "Powering the Future of AI-Enabled Medical Devices with NVIDIA Holoscan and RTI Connext," 2024, NVIDIA Developer. [Online]. Available: https://developer.nvidia.com/blog/powering-the-future-of-ai-enabled-medical-devices-with-nvidia-holoscan-and-rti-connext/?utm_source=chatgpt.com. [Accessed: Feb. 25, 2025].

[13] RTI, "RTI to Bring Data-Centric Connectivity to NVIDIA Holoscan with RTI Connext,"

2024. [Online]. Available: https://www.rti.com/news/nvidia-holoscan?utm_source=chatgpt.com. [Accessed: Feb. 26, 2025].

[14] L. Crawford, "NVIDIA Holoscan and RTI Connext Pave the Way for AI-Enabled Medical Devices," 2024. [Online]. Available: https://blockchain.news/news/nvidia-holoscan-rti-connext-ai-medical-devices?utm_source=chatgpt.com. [Accessed: Feb. 25, 2025].

[15] M. Grubis, "DDS in Patient Monitoring," GE Healthcare, 2020. [Online]. Available: https://www.dds-foundation.org/wp-content/uploads/2020/03/Application_Example-DDS_in_Patient_Monitoring_JB76285XX1_DDS-Foundation.pdf. [Accessed: Jan. 13, 2025].

[16] MGH MD PnP, "OpenICE – Open-Source ICE Reference Implementation," [Online]. Available: https://mdpnp.mgh.harvard.edu/astra-portfolio/openice/?utm_source=chatgpt.com. [Accessed: May. 26, 2025].

[17] Object Management Group, *DDS Security Specification, Version 1.2* (OMG Document No. formal/2024-06-01), 2024. [Online]. Available: https://www.omg.org/spec/DDS-SECURITY/1.2/.

[18] Real-Time Innovations (RTI), "RTI Security Plugins user manual," 2020. [Online]. Available: https://community.rti.com/static/documentation/connext-dds/6.1.2/doc/manuals/connext_dds_secure/users_manual/index.html#. [Accessed: Jul. 8, 2024].

[19] J. Du, C. Gao, and T. Feng, "Formal Safety Assessment and Improvement of DDS Protocol for Industrial Data Distribution Service," *Future Internet*, vol. 15, no. 1, pp. 24,

2023. [Online]. Available: https://doi.org/10.3390/fi15010024.

[20] X. Wang, H. Peng, J. Yin, H. Zhai, and C. Li, "Method for realizing DDS domain participant security authentication," U.S. Patent WO2021103431A1, 2021.

[21] R. S. Auliya, C. C. Chen, P. R. Lin, D. Liang, and W. J. Wang, "Optimization of message delivery reliability and throughput in a DDS-based system with per-publisher sending rate adjustment," *Telecommunication Systems*, vol. 84, no. 2, pp. 235-250, 2023. [Online]. Available: https://doi.org/10.1007/s11235-023-01045-x.

[22] Z. Kang, R. Canady, A. Dubey, A. Gokhale, S. Shekhar, and M. Sedlacek, "A Study of Publish/Subscribe Middleware Under Different IoT Traffic Conditions," in *Proc. 2020 Int. Workshop on Middleware and Applications for the Internet of Things (M4IoT)*, Delft, Netherlands, pp. 1–6. [Online]. Available: https://dl.acm.org/doi/10.1145/3429881.3430109.

[23] M. Takrouni, R. Bouhouch, and S. Hasnaoui, "Simulink Implementation of the Data Distribution Service for Vehicular Controllers on Top of GBE and AFDX," *The Computer Journal*, 2021. [Online]. Available: https://doi.org/10.1093/comjnl/bxaa176.

[24] A. Alaerjan, "Formalizing the Semantics of DDS QoS Policies for Improved Communications in Distributed Smart Grid Applications," *Electronics*, vol. 12, no. 10, p. 2246, 2023. [Online]. Available: https://doi.org/10.3390/electronics12102246.

[25] K. Peeroo, P. Popov, and V. Stankovic, "A Survey on Experimental Performance Evaluation of Data Distribution Service (DDS) Implementations," Centre for Software Reliability, City, University of London, Tech. Rep., Oct. 2023. [Online]. Available: https://arxiv.org/abs/2310.16630.

[26] V. Bode, D. Buettner, T. Preclik, C. Trinitis, and M. Schulz, "Systematic Analysis of DDS Implementations," in *Proc. 2023 ACM/IFIP Int. Middleware Conf. (Middleware '23)*, Bologna, Italy, pp. 234–246. [Online]. Available: https://dl.acm.org/doi/10.1145/3590140.3629118.

[27] Canonical Ltd., “Ubuntu: The open-source operating system,” [Online]. Available: https://ubuntu.com/download. [Accessed: Jan. 2, 2025].

[28] The OpenSSL Project, “OpenSSL: Cryptography and SSL/TLS Toolkit,” [Online]. Available: https://www.openssl.org/. [Accessed: Jan. 2, 2025].

[29] The Qt Company, “Qt Documentation,” [Online]. Available: https://doc.qt.io/. [Accessed: Jan. 5, 2025].

[30] Uwe Rathmann, “Qwt: Qt Widgets for Technical Applications,” [Online]. Available: https://qwt.sourceforge.io/. [Accessed: Jan. 8, 2025].

[31] Couchbase, Inc., “Couchbase Server,” [Online]. Available: https://www.couchbase.com/products/server. [Accessed: Jan. 9, 2025].

[32] Couchbase, Inc., “Couchbase Sync Gateway,” [Online]. Available: https://www.couchbase.com/products/sync-gateway. [Accessed: Jan. 9, 2025].

[33] Couchbase, Inc., “Couchbase Lite,” [Online]. Available: https://www.couchbase.com/products/lite. [Accessed: Jan. 8, 2025].

[34] Wireshark Foundation, “Wireshark: Network protocol analyzer,” [Online]. Available: https://www.wireshark.org/download.html. [Accessed: May. 25, 2024].

[35] RTI, “RTI Security Plugins Installation Guide,” 2022. [Online]. Available:

https://community.rti.com/static/documentation/connext-dds/6.1.2/doc/manuals/connext_dds_secure/installation_guide/index.htm. [Accessed: Sep. 14, 2024].