PQSec-DDS: Integrating Post-Quantum Cryptography into DDS Security for Robotic Applications

Javier Blanco-Romero
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
Madrid, Spain
frblanco@pa.uc3m.es

Vicente Lorenzo
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
Madrid, Spain
vilorenz@pa.uc3m.es

Florina Almenares
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
Madrid, Spain
florina@it.uc3m.es

Daniel Díaz-Sánchez
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
Madrid, Spain
dds@it.uc3m.es

Adrián Serrano Navarro
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
Madrid, Spain
100429115@alumnos.uc3m.es

***Abstract*—Leading cybersecurity agencies and standardization bodies have globally emphasized the critical need to transition towards Post-Quantum Cryptography (PQC) to defend against emerging quantum computing threats. They advocate PQC as a practical and cost-effective solution for security systems nowadays. Nevertheless, emerging technologies such as industrial systems, e.g., autonomous vehicles, air traffic management, diagnostic imaging machines, etc., and robotics systems, e.g., ROS2 (Robotic Operating System), have not started their evolution to enhance crypto-agility and security robustness. Some of these emerging technologies use the Data Distribution Service (DDS) standard as the underlying communication middleware protocol. DDS is a distributed publish-subscribe system that allows sending and receiving data by publishing and subscribing to topics across a network of connected nodes. However, DDS's security is based on traditional symmetric and asymmetric cryptography, which is vulnerable to quantum computing attacks. To address this issue, we propose the integration of PQC into DDS, through the development of a C/C++ library, called `pqsec-dds`, which can be integrated across different DDS implementations such as CycloneDDS or OpenDDS. A proof-of-concept demonstrates the viability of our approach in enhancing the security and crypto-agility of DDS-based systems.**

***Index Terms*—PQ, ROS2, Robotic Systems, IIoT, DDS**

Tipo de contribución: *Investigación original (límite 8 páginas)*

I. Introduction

Since the seminal work by Shor and Grover [1], [2], advancements in quantum computing are estimated to pose a threat to current cryptographic methods within the next 10 to 20 years [3]. This timeline suggests that without adaptation, traditional asymmetric encryption protocols will become ineffective against quantum computing capabilities. To overcome this threat the usage of post-quantum cryptography (PQC) or quantum-resistant algorithms is proposed and is being implemented [4].

Leading cybersecurity agencies globally, including the American NCSC (National Cybersecurity and Communications Integration Center) [5], NSA (National Security Agency) [6], CISA (Cybersecurity and Infrastructure Security Agency), NIST (National Institute of Standards and Technology) [7], and European bodies such as ENISA (European Union Agency for Network and Information Security), or national agencies in European countries such as CCN ("Centro Criptológico Nacional", Spanish National Cybersecurity Center) [8], ANSSI ("Agence Nationale de la Sécurité des Systèmes d'Information", National Agency for the Security of Information Systems), BSI ("Bundesamt für Sicherheit in der Informationstechnik", Federal Office for Information Security), NLNCSA (National Cyber Security Centre), and the Swedish NCS (National Computer Security) Centre [9], have collectively emphasized the critical need to transition towards PQC to defend against the emerging quantum computing threat. These bodies and agencies advocate for quantum-resistant cryptography as a more practical, cost-effective solution, emphasizing the need for further research and development to address the current shortcomings of QKD [6][9]. Thus, a proactive approach in preparing for the PQC transition involves the development of quantum-readiness roadmaps, collaboration with vendors, prioritization of key assets in migration plans, and integration of PQ algorithms into existing cryptographic frameworks and security protocols, among others [7].

The integration of post-quantum cryptography into communication security protocols such as TLS (Transport Layer Security) and DTLS (Datagram TLS) is a key area of development, aimed at bolstering protocol security. For that, the IETF (Internet Engineering Task Force) is working on integrating PQ in TLS 1.3 [10]. This effort extends the potential for enhanced security measures to any protocol relying on TLS or DTLS foundations. Adoption of post-quantum security measures is being facilitated through both open-source [11] and commercial [12] initiatives, reflecting a broad commitment to evolving TLS and DTLS against quantum threats. This is being mainly tested and used for Internet applications, but its integration in emerging and important fields such as IoT (Internet of Things) or IIoT (Industrial IoT) is absent or still in its early stages. These fields involve applications and systems such as robotics, healthcare, energy, defense, transportation, and industrial automation, e.g., SCADA systems [13]. The impact of unauthorized access or data modification within these applications could be quite severe [14].

In recent years, the field of robotics has witnessed significant advancements, leading to the widespread adoption of robotic systems in various domains. However, as these systems become more interconnected and autonomous, the need for robust cybersecurity measures has become increasingly apparent. The potential vulnerabilities in robotic systems, such as insecure communication channels, weak authentication mechanisms, and outdated software components, have raised concerns among researchers and industry professionals alike [15], [16], [17]. Numerous studies have emphasized the importance of improving encryption, authorization, and authentication techniques to mitigate cyber security risks [18], [19], [20]. Researchers have explored various approaches to enhance the security of robotic systems, such as hardening the Robot Operating System (ROS), using symmetric encryption algorithms and semantic rules [21], adopting standardized operating systems and formalizing authentication methods [18], and addressing issues related to outdated software components and weak authentication schemes [19], [22].

Despite these growing concerns and efforts, little attention has been paid to the specific threat posed by quantum computing. In this context, we focus on the Data Distribution Service protocol (DDS), as a standardized and fundamental middleware protocol for IIoT, that faces parallel challenges. DDS facilitates real-time data exchange in distributed systems through a data-centric publish-subscribe model, supporting both UDP and TCP transports for network communication. Its importance is further highlighted by its recent integration as the communication middleware in ROS 2, addressing the limitations of ROS 1 in terms of real-time operations, timely communication, and scalability in large-scale distributed systems. Nevertheless, DDS architecture requires a tailored approach to integrate quantum-resistant security. Its security specification defines a Service Plugin Interface (SPI) framework, covering aspects such as authentication, encryption, and access control to ensure secure data exchange [23].

To address the need for post-quantum security in DDS-based systems, we develop a plugin, PQSec-DDS [24], that integrates post-quantum cryptography into the DDS security framework. By leveraging the SPI provided by DDS, our plugin enhances the authentication mechanism with quantum-resistant algorithms. The PQSec-DDS plugin is designed to be compatible with C and C++ implementations such as CycloneDDS and OpenDDS.

The remainder of this paper is organized as follows. Section II introduces the basis and main concepts about DDS and PQC, as well as describing briefly related work. In section III, we provide details on how to integrate post-quantum cryptography into the DDS authentication handshake and an overview of our proposed PQSec-DDS framework. Finally, Section IV concludes the paper and outlines future research directions.

II. Background & Related Work

This section introduces the main concepts and technical issues of DDS and PQC.

A. Data Distribution Service (DDS)

1) DDS Overview: The Data Distribution Service is a middleware protocol developed by the Object Management Group (OMG) for data-centric publish-subscribe communication in distributed systems. It facilitates the exchange of data among software components distributed across networked computers to act as a unified system. DDS aims for interoperability across different vendors' implementations through the DDS Interoperability Wire Protocol (DDSI-RTPS, Real-Time Publish-Subscribe Protocol), ensuring applications based on DDS can work together using IP multicast. This standard supports real-time, scalable, and reliable data distribution, essential for systems requiring robust data exchange mechanisms [25][26]. So this middleware protocol is being used by robotic systems such as ROS2, and other IoT systems.

RTPS, as a standard protocol approved by the IEC within the Real-Time Industrial Ethernet Suite (IEC-PAS-62030), aligns with DDS's requirements for discovery, fault tolerance, reliability, and timeliness, without central points of failure. Originally specified for UDP/IP, RTPS has been extended by some vendors to include TCP/IP support, broadening its usability across various network configurations.

RTPS is a decentralized protocol capable of running over multicast and connectionless best-effort transports like UDP/IP. It supports unidirectional data exchange, where publishers "push" data updates to the local caches of co-located subscribers, regulated by Quality of Service (QoS) contracts. Key technical features of RTPS include:

- Built-in discovery service for dynamic discovery and monitoring of publishers and subscribers without centralized name servers.
- Fault tolerance and decentralized architecture without single points of failure.
- Extensibility and backward compatibility, allowing protocol evolution while maintaining interoperability with deployed systems.
- Configurability for balancing reliability and timeliness requirements per data delivery.
- Scalability to support large-scale networks with potentially thousands of participants.
- Type-safety to prevent programming errors from compromising remote node operations.

2) DDS Security: Initially, DDS security relied on TLS or DTLS protocols for ensuring data integrity and confidentiality, given the lack of a specific security framework within the DDS standards [27] [14]. DDS-compliant products, such as RTI DDS, OpenSplice DDS, and eProsima Fast DDS, have incorporated security mechanisms based on these protocols. For example, RTI DDS uses DTLS to encapsulate notifications, while OpenSplice DDS leverages domain partitioning to implement access control. eProsima Fast DDS and RTI Connext also support securing TCP transports with TLS [28] [29]. However, TLS is not standardized across all DDS implementations [30].

The limitations of these approaches, particularly in supporting multicast due to DTLS's inherent client/server structure, highlighted the need for a comprehensive DDS security specification. Soroush et al. studied the security of DDS on top of secure TLS/DTLS transports and compared it with RTI's beta version of the DDS Security Specification [31].

The DDS-Security specification [23] extends DDS by introducing predefined security features through a Service Plugin Interface framework, which supports multicast and real-time communication. DDS-Security works over any transport protocol with configurable Quality of Service (QoS) settings. Likewise, it offers interoperability across vendor implementations. It introduces five key SPIs:

- **Authentication:** Ensures the identity authentication of domain participants through a trusted identity CA. The builtin Authentication plugin, `DDS:Auth:PKI-DH`, uses RSA or ECDSA signature algorithms (with key sizes of 2048-bit and 256-bit NIST P-256 curve, respectively) and DHE or ECDHE as key exchange methods, involving a 3-message handshake protocol.
- **Cryptography:** Manages encryption, signing, and hashing for data integrity. The builtin Cryptography plugin, `DDS:Crypto:AES-GCM-GMAC`, provides encryption and message authentication using AES in Galois Counter Mode (AES-GCM) and Galois Message Authentication Code (AES-GMAC).
- **Access Control:** Regulates permissions for DDS operation execution, offering finer-grained control and allowing different permissions for different applications within a DDS domain.
- **Logging:** Facilitates auditing of security-related events. Not universally required for compliance, leading to partial implementation across DDS systems.
- **Data Tagging:** Allows for the addition of metadata tags to data samples. Similarly, not a universal requirement for compliance.

These security features are designed to protect against unauthorized subscriptions, unauthorized publication, tampering, replay attacks, and unauthorized access to data, providing a more comprehensive and standardized approach to securing DDS-based systems. The selection of these built-in plugins was driven by key functional and non-functional requirements, including scalable performance, robustness, ease of use, and leveraging existing security infrastructure, while aiming to fit the data-centric DDS model and avoid centralized components that could become single points of failure.

3) DDS Authentication Handshake: The DDS authentication handshake is similar to the TLS handshake [32] [23]. The authentication process begins with a 3-message handshake initiated by the participant with the lower unique identifier (GUID). Let's say Participant Bob wants to start communicating with Participant Alice.

Bob calls `begin_handshake_request()` which sends its Diffie-Hellman public value (B_{dh}), its certificate (B_{cert}) signed by the Identity CA, the domain permissions document signed by the Permissions CA, and a random 256-bit nonce (B_{rand}) for authentication. Alice calls `begin_handshake_reply()` to validate B_{cert} against the Identity CA. It responds with its own DH public value (A_{dh}), certificate (A_{cert}), permissions document, the received B_{rand}, a new 256-bit nonce (A_{rand}), and a signature (A_{sig}) over the nonces. Bob calls `process_handshake()` to validate A_{cert} and A_{sig}. It sends back its own signature (B_{sig}) over the nonces. Alice finalizes with `process_handshake()`, verifying B_{sig}. Both use the exchanged DH public values to compute a shared secret with `get_shared_secret()` which is hashed with SHA256. Bob generates a master salt, session ID, and IV suffix. Using HKDF, Bob derives a master sender key from the shared secret, salts, and nonces. This key is encrypted with the shared secret and sent to Alice over a secure channel.

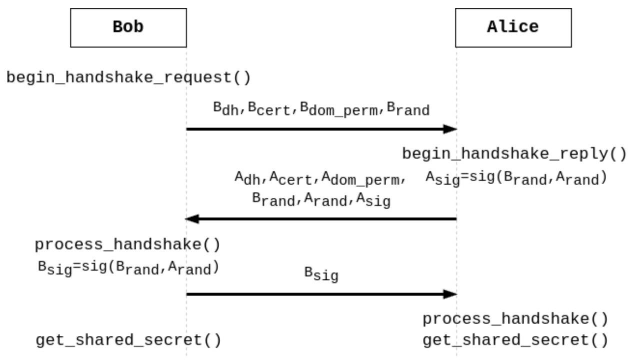

Fig. 1. DDS Authentication handshake with mutual authentication between two domain participants, Alice and Bob. We omit the discovery mechanism of the participants.

Both expand the master key with the salts, session ID, and nonces into a session key using HKDF with HMAC-SHA256. The IV comprises the session ID and suffix. The session key and IV secure future communications, with the IV incremented for each operation under this session key.

The DDS handshake is depicted in Figure 1

4) DDS Implementations and Security: Table I provides an overview of the main DDS implementations, highlighting their licensing, programming language, key exchange mechanisms, and signature algorithms. In what follows, we have considered implementations with open licenses, instead of commercial software.

The DDS implementations considered in this study have their own DDS Security integration, as described below.

a) Cyclone DDS: Cyclone DDS allows dynamic loading of external security plugins for authentication, cryptography, and access control (see *External Plugin Development* [36]). The security plugin API consists of a few header files, with separate headers for each plugin type, such as `dds_security_api_authentication.h`, `dds_security_api_cryptography.h`, and `dds_security_api_access_control.h`. The API functions and types are prepared from the IDL by adding the `DDS_Security_` namespace prefix, and separate `DDS_Security_` data types are defined instead of extending DDS built-in topic data types.

To configure external security plugins in Cyclone DDS, developers can use an XML configuration file. By creating an XML file with the desired settings for the authentication, cryptography, or access control plugins and setting the `CYCLONEDDS_URI` environment variable to point to the XML configuration file path, the plugins can be easily integrated into the DDS system.

TABLE I
CONSIDERED DDS IMPLEMENTATIONS

Implementation	License	Language	Key Exchange	Signature
eProsima Fast DDS [33]	Apache 2	C++	DH_2048_256 ECDH_prime256v1	RSA_SHA256 or ECDSA_SHA256
Eclipse Cyclone DDS [34]	Eclipse Public License v2.0	C	DH+MODP-2048-256 or ECDH+prime256v1-CEUM	RSASSA-PSS-SHA256 or ECDSA-SHA256
OpenDDS [35]	Open Source	C++	DH_2048_MODP_256_PRIME_STR or ECDH_PRIME_256_V1_CEUM	RSASSA-PSS-SHA256 or ECDSA-SHA256
RTI Connext DDS	Commercial, Research	C and C++	DHE_MODP_2048_256 or ECDHE_CEUM_P256/P384	RSASSA_PSS or ECDSA_P256_SHA256 or ECDSA_P384_SHA384
GurumNetworks GurumDDS	Commercial	C++	-	-

b) OpenDDS: OpenDDS incorporates its security mechanisms, such as authentication functionality and credential management, within the `dds/DCPS/security/` directory. It is possible to implement custom plugins, although its documentation is not as clear as CycloneDDS. Developers can write custom plugins in C+ (the optional support for external C plugins of the DDS Security v1.1 specification is not yet implemented) [37]. The plugin is then built and integrated into OpenDDS at the application level, loading the plugin and creating a custom security configuration.

c) eProsima Fast DDS: An examination of the Fast-DDS documentation and source code reveals that this implementation is not yet compatible with external plugins. As a result, eProsima Fast DDS will not be considered further in this study.

B. Post-Quantum Cryptography

1) PQC Overview: PQC -also known as quantum-resistant cryptography- standardization is yet open. The algorithms use public key schemes for Key Encapsulation Mechanism (KEM) and signature. KEM defines schemes for key exchange and encryption to transport a key from one party to another [38]. Signature scheme defines both generation and verification of digital signatures [39].

The algorithms proposed so far are mainly based on five schemes: lattices, multivariate polynomials, error-correcting and error-detecting codes, hash-based signatures, and isogeny of elliptic curves. Some lattice-based algorithms have been standardized by NIST, such as Kyber [40] for KEM and Dilithium [41] for signature. In addition, one algorithm based on stateless hash has been also standardized [42], called SPHINCS+.

These algorithms differ significantly in key sizes and encrypted or signed text compared to traditional algorithms. Likewise, new algorithms can be standardized. Therefore, algorithms can be used according to the specific requirements of each scenario. The goal is to have a crypto-agile API that enables seamless integration of these algorithms into applications regardless of cryptographic material size, to ensure long-term security.

2) Integrating Post-Quantum Cryptography in TLS/DTLS: TLS 1.2 [43] and its datagram counterpart DTLS 1.2 [44] leverage the TLS handshake for peer authentication, algorithm negotiation, and shared secret computation, followed by the TLS record protocol for data confidentiality and integrity using symmetric cryptography with the established keys. These protocols rely on X.509 certificates, cipher suites specifying cryptographic algorithms, and key exchange methods like Diffie-Hellman. The handshake involves a ClientHello/ServerHello exchange with nonces, certificate/signature exchange, and Diffie-Hellman-based key derivation using HMAC. DTLS 1.2 introduces reliability mechanisms like sequencing and retransmissions to adapt TLS 1.2 to unreliable transports like UDP.

The more recent TLS 1.3 [45] and DTLS 1.3 [46] specifications introduce significant improvements, enhancing security and performance. They use only modern, secure cryptographic algorithms, removing obsolete ciphers. The handshake process is streamlined, reducing round trips for connection establishment. TLS 1.3 incorporates advancements like Elliptic Curve Diffie-Hellman (ECDHE) for forward-secret key exchange, ChaCha20-Poly1305 authenticated encryption, and a revised key derivation process based on the HKDF scheme. DTLS 1.3 closely follows TLS 1.3 while maintaining datagram-specific reliability mechanisms.

Traditional public-key operations in the TLS handshake can be replaced with post-quantum primitives [47], [48], [49]. RSA/ECDSA signatures can be substituted by post-quantum signature schemes. For TLS 1.3, the Diffie-Hellman key exchange can be replaced by using the server's key encapsulation mechanism encapsulation operation and sending the ciphertext instead of the server's public key, as demonstrated by Bos et al. [47]. Their work includes a proof of the security of replacing Diffie-Hellman with KEM key exchange in TLS 1.2. The proposed approach for TLS 1.3 leverages hybrid primitives, combining post-quantum algorithms with traditional pre-quantum schemes like ECDH [10]. As breaking a hybrid scheme requires compromising both components, this approach allows confidence in the implementation and security of the pre-quantum scheme against classical attacks to carry over to the hybrid scheme, providing quantum resistance while maintaining pre-quantum security assurances during the transition period.

In recent years, the integration of post-quantum cryptography into widely-used software libraries and protocols has gained significant attention. One notable example is the Open Quantum Safe project's fork of OpenSSL [11], which incorporates quantum-resistant algorithms for key exchange and digital signatures into the widely-used TLS 1.2 and 1.3 protocols. This integration leverages the liboqs library [50], which provides a collection of post-quantum cryptographic algorithms. Additionally, oqs-provider, an OpenSSL 3 provider, has been developed to integrate post-quantum cryp-

Requirement	DDS	TLS/DTLS 1.2	TLS/DTLS 1.3 (Default)
PKI with X.509 Certificates	Yes	Yes	Yes
Signature Algorithm	RSA or ECDSA	RSA or ECDSA	RSA or ECDSA
Key Exchange Method	DHE or ECDHE	DHE or ECDHE	ECDHE
Encryption	AES-GCM	AES-GCM	AES-GCM
Integrity	AES-GMAC	AES-GMAC	AES-GMAC
Hybrid Mode	Not Specified	Not Specified	Supported (Post-Quantum + Classical)

TABLE II
COMPARISON OF SECURITY PROTOCOLS: DDS, TLS/DTLS 1.2, AND TLS/DTLS 1.3 WITH DEFAULT AND POST-QUANTUM CRYPTOGRAPHY INTEGRATION

tography through the liboqs library [51], further expanding the availability of quantum-resistant algorithms in the OpenSSL ecosystem. Another significant development is the support for post-quantum cryptography in the DTLS 1.3 protocol by wolfSSL [12], which enables the use of quantum-resistant algorithms in datagram-based secure communication. These initiatives demonstrate the growing effort to prepare critical software infrastructure for the potential advent of large-scale quantum computers and the associated security risks.

III. PQSec-DDS: Integrating Post-Quantum Cryptography in DDS Authentication Handshake

To address the need for post-quantum security in DDS-based systems, we propose PQSec-DDS, a plugin that integrates post-quantum cryptography into the DDS security framework. Our plugin focuses on the Authentication plugin, which is one of the five Service Plugin Interfaces (SPIs) defined in the DDS security specification. The Authentication plugin directly performs asymmetric or public key cryptography operations, and the Access Control and Cryptographic plugins rely on it for operations dependent on public key cryptography, such as using the shared secret for deriving key material or identity handles for access control.

PQSec-DDS integrates post-quantum signature and key encapsulation mechanisms through the liboqs library and the oqs-provider for OpenSSL, mirroring the integration of post-quantum cryptography in protocols like TLS and DTLS. We have made the source code of our plugin publicly available on GitHub [24] (work in progress).

A. Integrating Post-Quantum Cryptography in DDS Authentication Handshake

The integration of PQ cryptography starts replacing public keys in DH with the PQ equivalents as described in [47], [48], [49]. Thus, the initiator Bob calls `begin_handshake_request()` sending a KEM public key B_{pk} generated by `KEM.KeyGen()`, retaining the secret key B_{sk}. His certificate, permissions, and nonce B_{rand} are also sent. Alice calls `begin_handshake_reply()`, running `KEM.Encaps()` on B_{pk} to get an encapsulated ciphertext A_{ct} and shared secret A_{ss}. She sends A_{ct}, A_{cert}, permissions, B_{rand}, a new A_{rand}, and a signature A_{sig} over the nonces using a post-quantum signature scheme. After validating A_{cert} and A_{sig}, Bob calls `process_handshake()`, running `KEM.Decaps(`A_{ct}, B_{sk}`)` to retrieve the shared B_{ss}, sending back B_{sig} over the nonces. Finally, Alice calls `process_handshake()` verifying B_{sig}, with both retrieving the `get_shared_secret()` for future key derivation. The PQ integration into the DDS Authentication handshake is depicted in Figure 2.

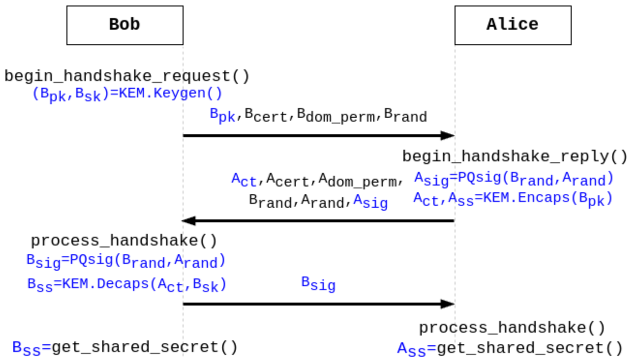

Fig. 2. Integration of Post-Quantum KEMs and signatures into DDS Authentication handshake with mutual authentication. We omit the discovery mechanism of domain participants. The required changes are highlighted in blue.

The PQ key exchange and the handling of certificates and public keys can be done using two approaches: either directly using the OpenSSL library endowed with the liboqs provider via the EVP and X509 APIs, or using the liboqs library directly.

The key exchange impacts the `begin_handshake_request()`, `begin_handshake_reply()`, and `process_handshake()` functions, where the classical DH key generation and shared secret derivation are replaced with PQ KEMs.

On the other hand, the handling of certificates affects functions like `get_certificate_contents()`, `verify_certificate()`, and the signature creation and verification processes in `begin_handshake_reply()` and `process_handshake()`. These functions rely on OpenSSL's `X509` and `EVP` APIs, which can work with post-quantum algorithms through the oqs-provider.

It's important to note that while the core of the PQ integration is in the handshake process, several other functions in the DDS Security plugin may require minor adjustments to accommodate the new post-quantum algorithms, such as adapting key encoding and decoding functions to handle the different key sizes of PQ schemes.

B. PQSec-DDS Design

The PQSec-DDS Authentication plugin is designed to be compliant with the conformance points outlined in the DDS Security specification, with extended support of PQC algorithms into the DDS security framework. The plugin targets both C and C++ API compatibility, ensuring portability across compliant DDS implementations, leveraging the liboqs library and the Open Quantum Safe's oqs-provider to provide post-quantum key encapsulation mechanisms and signature schemes. The architecture of the plugin allows it to interact with the DDS middleware, as depicted in Figure 3. The figure depicts the DDS middleware architecture in the bottom layer, which includes a module `DDS-Security` for RTPS and DDS. This module interacts with the plugin layer, in our case, the PQSec-DDS Authentication plugin, which incorporates the functions associated with the signatures and certificates handling and key exchange modules. The plugin uses liboqs and OpenSSL libraries as underlying libraries to implement the required functions.

The integration of post-quantum security in both CycloneDDS and OpenDDS follows a similar approach. The common steps involve importing/generating the necessary API header files, implementing the DDS Security Authentication functions using the post-quantum cryptography algorithms -provided by liboqs and OpenSSL-, and building the plugin as a dynamic library that can be loaded by the respective DDS implementations.

For CycloneDDS, the plugin development process involves importing the required API header files, such as `dds_security_api_authentication.h`, and implementing the DDS Security Authentication functions, such as initializing the authentication process, handling handshakes, and managing identities. The plugin leverages liboqs and its OpenSSL provider to perform key encapsulation and signature operations, ensuring the security of the authentication process in a post-quantum scenario. Once built as a dynamic library, the plugin can be loaded by CycloneDDS using the .xml configuration file.

Similarly, for OpenDDS, the integration of post-quantum security involves defining the plugin interface using IDL, generating C++ header from the IDL using the `tao_idl` tool, implementing the plugin in C++ by inheriting from the generated plugin interface class, and implementing the required functions according to the post-quantum cryptography requirements. The plugin is then built and integrated into OpenDDS at the application level, loading the plugin and creating a custom security configuration.

IV. Conclusions and Future Work

In this paper, we have proposed the integration of post-quantum cryptography into the DDS security specification, focusing on the authentication handshake. Our findings suggest that the integration process closely mirrors that of TLS/DTLS, indicating that the lessons learned from the post-quantum transition in these protocols can be applied to DDS.

One of the key advantages of the DDS Security Specification SPI is its API design, which facilitates the creation of external plugins. This feature enables crypto-agility, a crucial requirement in the transition to post-quantum cryptography. However, it is important to note that not all open-source DDS vendors currently provide a straightforward way to integrate external plugins, which may present challenges for some implementations.

As future work, we plan to benchmark the integration of post-quantum cryptography into Cyclone DDS and OpenDDS, within the context of ROS2. This will provide valuable insights into the performance and practicality of post-quantum cryptography in robotic systems.

Overall, our research highlights the importance of preparing for the post-quantum era in robotic systems that rely on DDS for secure communication. By integrating post-quantum cryptography into the DDS Security Specification, we can ensure that these systems remain secure against potential quantum computing threats in the future.

Acknowledgments

This work was supported by the Spanish Government under the grant TED-2021-130369B-C32 funded by MICIU/AEI/ 10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR" and the research project "Enhancing Communication Protocols with Machine Learning while Protecting Sensitive Data (COMPROMISE)" PID2020-113795RB-C32, funded by MICIU/AEI/10.13039/501100011033. In addition, it was partially supported by project i-SHAPER, which is being carried out within the framework of the Recovery, Transformation, and Resilience Plan funds, funded by the European Union (Next Generation).

References

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.
[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.
[3] M. Mosca and M. Piani, "Global Risk Institute: Quantum Threat Timeline Report 2022," https://globalriskinstitute.org/publication/2022-quantum-threat-timeline-report/, 2022, accessed: 6 march 2024.
[4] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.
[5] National Cyber Security Centre (NCSC), "Quantum security technologies," https://www.ncsc.gov.uk/pdfs/whitepaper/quantum-security-technologies.pdf, Mar 2020, accessed: 6 march 2024.
[6] National Security Agency (NSA), "Quantum key distribution (qkd) and quantum cryptography (qc)," https://www.nsa.gov/Cybersecurity/Quantum-Key-Distribution-QKD-and-Quantum-Cryptography-QC/, accessed: 2024-02-15.
[7] N. CISA and NIST, "Quantum-readiness: Migration to post-quantum cryptography," https://media.defense.gov/2023/Aug/21/2003284212/-1/-1/0/CSI-QUANTUM-READINESS.PDF, Aug 2023, accessed: 15 february 2024.
[8] Centro Criptológico Nacional (CCN)-PYTEC, "Recomendaciones para una transición postcuántica segura," December 2022, cCN-TEC 009.
[9] F. C. A. (ANSSI), F. O. for Information Security (BSI), N. N. C. S. A. (NLNCSA), and S. A. F. Swedish National Communications Security Authority, "Position paper on quantum key distribution," Tech. Rep., January 2024.
[10] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid key exchange in tls 1.3," Internet-Draft, Internet Engineering Task Force (IETF) Network Working Group, Tech. Rep., Sep 2023, draft-Ietf-Tls-Hybrid-Design-09. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/09/
[11] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in *Selected Areas in Cryptography (SAC) 2016*, ser. Lecture Notes in Computer Science, vol. 10532. Springer, oct 2017, pp. 1–24. [Online]. Available: https://openquantumsafe.org
[12] wolfSSL, "DTLS 1.3 support for Post-Quantum Cryptography," https://www.wolfssl.com/dtls-1-3-support-post-quantum-cryptography/, wolfSSL, January 2023, do you want to start using wolfSSL's DTLS 1.3 implementation?
[13] Real-Time Innovations, "DDS: the Right Middleware for the Industrial Internet of Things?" Real-Time Innovations, Inc., Tech. Rep., October 2014, https://info.rti.com/hubfs/whitepapers/Right_Middleware_for_IIoT.pdf.

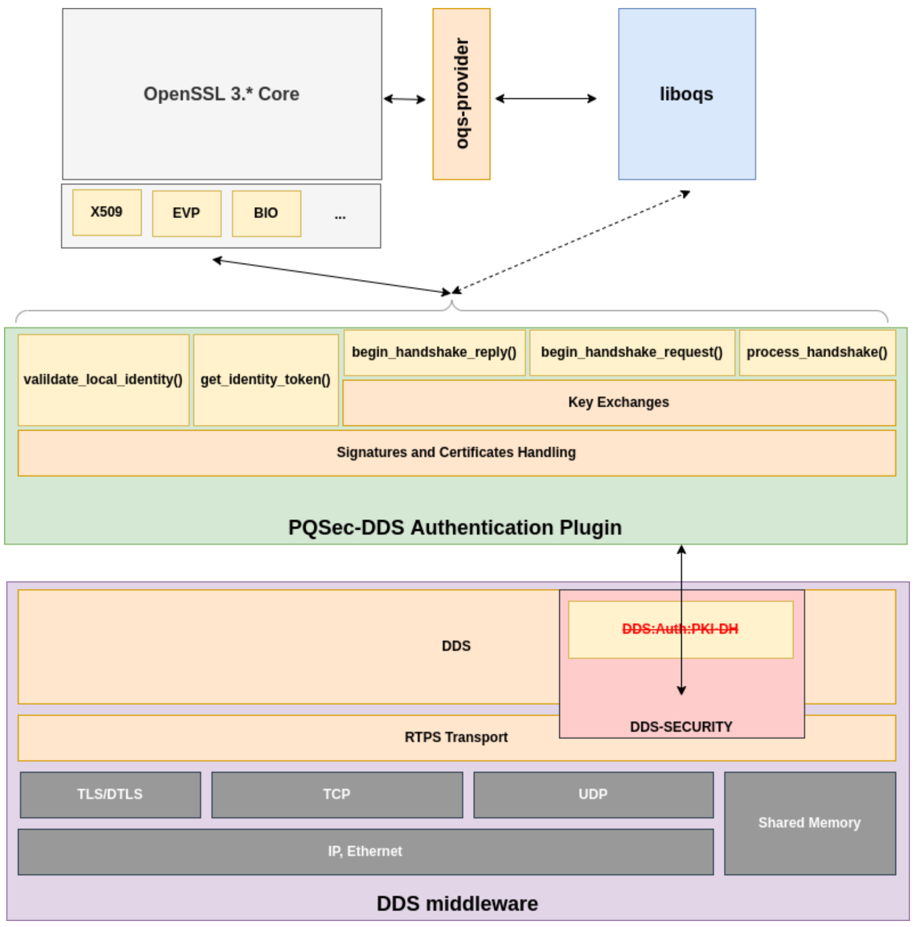

Fig. 3. Architecture of the PQSec-DDS Authentication plugin and its interaction with the DDS middleware and the liboqs and OpenSSL libraries.

[14] T. White, M. N. Johnstone, and M. Peacock, "An investigation into some security issues in the dds messaging protocol," 2017.

[15] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Robotics and Autonomous Systems*, vol. 98, pp. 192–203, 2017.

[16] V. Dutta and T. Zielińska, "Cybersecurity of robotic systems: Leading challenges and robotic system design methodology," *Electronics*, vol. 10, no. 22, p. 2850, 2021.

[17] A. Botta, S. Rotbei, S. Zinno, and G. Ventre, "Cyber security of robots: A comprehensive survey," *Intelligent Systems with Applications*, p. 200237, 2023.

[18] G. W. Clark, M. V. Doran, and T. R. Andel, "Cybersecurity issues in robotics," in *2017 IEEE conference on cognitive and computational aspects of situation management (CogSIMA)*. IEEE, 2017, pp. 1–5.

[19] F. Maggi, D. Quarta, M. Pogliani, M. Polino, A. M. Zanchettin, and S. Zanero, "Rogue robots: Testing the limits of an industrial robot's security," *Trend Micro, Politecnico di Milano, Tech. Rep*, pp. 1–21, 2017.

[20] H. D. Elikhchi, T. Hamid, and M. Akpoduado, "Robotics cyber security issues," in *International Conference on Advances in Communication Technology and Computer Engineering*. Springer, 2023, pp. 217–225.

[21] J. Balsa-Comerón, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, C. Fernández-Llamas, and V. Matellán-Olivera, "Cybersecurity in autonomous systems: hardening ros using encrypted communications and semantic rules," in *ROBOT 2017: Third Iberian Robotics Conference: Volume 2*. Springer, 2018, pp. 67–78.

[22] G. Lacava, A. Marotta, F. Martinelli, A. Saracino, A. La Marra, E. Gil-Uriarte, and V. M. Vilches, "Cybsersecurity issues in robotics." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 12, no. 3, pp. 1–28, 2021.

[23] "DDS Security Version 1.1," https://www.omg.org/spec/DDS-SECURITY/1.1/About-DDS-SECURITY, Object Management Group, Inc., July 2018, oMG Document Number: formal/2018-04-01.

[24] J. Blanco-Romero and A. Serrano Navarro, "Pqsec-dds github repository," 2024, accessed: 2024-03-27. [Online]. Available: https://github.com/qursa-uc3m/pqsec-dds

[25] "Data Distribution Service Version 1.4," https://www.omg.org/spec/DDS/1.4/, Object Management Group, Inc., March 2015, describes the Data-Centric Publish-Subscribe (DCPS) model for efficient and robust data delivery.

[26] "DDS Interoperability Wire Protocol Version 2.5," https://www.omg.org/spec/DDSI-RTPS/2.5/, Object Management Group, Inc., April 2022, describes the Real-Time Publish Subscribe Protocol for DDS interoperability.

[27] C. Esposito and M. Ciampi, "On security in publish/subscribe services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 966–997, 2014.

[28] eProsima, *6.7. TLS over TCP — Fast DDS 2.13.1 documentation*, 2023, accessed: 2024-02-15. [Online]. Available: https://fast-dds.docs.eprosima.com/en/latest/fastdds/transport/tcp/tls.html

[29] Real-Time Innovations, Inc., "RTI TLS Support Release Notes, Version 7.2.0," Real-Time Innovations, Inc., Tech. Rep., October 2023, © 2010-2023 Real-Time Innovations, Inc. All rights reserved.

[30] *RTI Security Plugins User's Manual: Choosing the Right Technology to Protect Your Data*, Real-Time Innovations, accessed: 6 march 2024. [Online]. Available: https://community.rti.com/static/documentation/connext-dds/current/doc/manuals/connext_dds_secure/users_manual/p1_welcome/overview.html#section-choosing-the-right-technology-to-protect-your-data

[31] H. Soroush, D. Arney, and J. Goldman, "Toward a safe and secure medical internet of things," *IIC J. Innov*, vol. 2, no. 1, pp. 4–18, 2016.

[32] M. Friesen, G. Karthikeyan, S. Heiss, L. Wisniewski, and H. Trsek, "A comparative evaluation of security mechanisms in dds, tls and dtls," in *Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu 2018*. Springer Berlin Heidelberg, 2020, pp. 201–216.

[33] eProsima, "Fast dds github repository," 2024, accessed: 2024-03-27. [Online]. Available: https://github.com/eProsima/Fast-DDS

[34] E. Foundation, “Cyclonedds github repository,” 2024, accessed: 2024-03-27. [Online]. Available: https://github.com/eclipse-cyclonedds/cyclonedds
[35] Object Computing, Inc., “Opendds github repository,” 2024, accessed: 2024-03-27. [Online]. Available: https://github.com/OpenDDS/OpenDDS
[36] Eclipse Cyclone DDS, “DDS Security - External Plugin Development,” https://cyclonedds.io/docs/cyclonedds/0.8.2/security.html#external-plugin-development, 2022, accessed: 2024-03-12.
[37] OpenDDS Foundation, “DDS Security Implementation Status,” https://opendds.readthedocs.io/en/dds-3.27/devguide/dds_security.html#dds-security-implementation-status, OpenDDS Foundation, 2023, accessed: 2024-03-12.
[38] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon, “NIST SP 800-56B Rev. 2. Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography,” 2019.
[39] NIST, “FIPS 186-5. Digital Signature Standard (DSS),” 2023.
[40] National Institute of Standards and Technology, “Draft fips (federal information processing standards) 203, module-lattice-based key-encapsulation mechanism standard,” NIST, Information Technology Laboratory, Tech. Rep., 2003.
[41] ——, “Draft fips (federal information processing standards) 204, module-lattice-based digital signature standard,” NIST, Information Technology Laboratory, Tech. Rep., 2003.
[42] ——, “Draft fips (federal information processing standards) 205, stateless hash-based digital signature standard,” NIST, Information Technology Laboratory, Tech. Rep., 2003.
[43] T. Dierks and E. Rescorla, “The transport layer security (tls) protocol version 1.2,” IETF, RFC 5246, Aug 2008. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5246.txt
[44] E. Rescorla and N. Modadugu, “Datagram transport layer security version 1.2,” IETF, RFC 6347, Jan 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6347.txt
[45] E. Rescorla, “The transport layer security (tls) protocol version 1.3,” IETF, RFC 8446, 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8446.txt
[46] E. Rescorla, H. Tschofenig, and N. Modadugu, “The datagram transport layer security (dtls) protocol version 1.3,” IETF, RFC 9147, Apr 2022, obsoletes RFC 6347. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9147.txt
[47] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, “Post-quantum key exchange for the tls protocol from the ring learning with errors problem,” in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 553–570.
[48] K. Bürstinghaus-Steinbach, C. Krauß, R. Niederhagen, and M. Schneider, “Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls,” in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 841–852.
[49] T. Wiggers, “Post-quantum tls,” Ph.D. Thesis, Radboud University, 2024.
[50] Open Quantum Safe Project, “liboqs: Library for quantum-resistant cryptographic algorithms,” https://github.com/open-quantum-safe/liboqs, 2024, accessed: 2024-03-21.
[51] ——, “OpenSSL 3 provider containing post-quantum algorithms,” https://github.com/open-quantum-safe/oqs-provider, 2024, accessed: 2024-03-21.