DDS Security+: Enhancing the Data Distribution Service With TPM-based Remote Attestation

Paul Georg Wagner
Pascal Birnstill
paul-georg.wagner@iosb.fraunhofer.de
pascal.birnstill@iosb.fraunhofer.de
Fraunhofer IOSB
Karlsruhe, Germany

Jürgen Beyerer
juergen.beyerer@iosb.fraunhofer.de
Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

The Data Distribution Service (DDS) is a widely accepted industry standard for reliably exchanging data over the network using a publish-subscribe model. While DDS already includes basic security features such as participant authentication and access control, the possibilities of leveraging Trusted Platform Modules (TPMs) to increase the security and trustworthiness of DDS-based applications have not been sufficiently researched yet. In this work, we show how TPM-based remote attestation can be effectively integrated into the existing DDS security architecture. This enables application developers to verify the code integrity of remote DDS participants during the operation of the distributed system. Our solution transparently extends the DDS secure channel handshake, while cryptographically binding the established communication channels to the attested software stacks. We show the security properties of our proposal by formally verifying the resulting remote attestation protocol using the Tamarin theorem prover. We also implement our solution as a fork of the popular eProsima FastDDS library and evaluate the resulting performance impact when conducting TPM-based remote attestations of DDS applications.

CCS CONCEPTS

• **Security and privacy → Security protocols**; **Embedded systems security**; **Distributed systems security**; • **Computer systems organization → Embedded and cyber-physical systems**; **Distributed architectures**.

KEYWORDS

Data Distribution Service, DDS Security, Remote Attestation, Trusted Platform Modules, TPM, Integrity Measurement

ACM Reference Format:
Paul Georg Wagner, Pascal Birnstill, and Jürgen Beyerer. 2024. DDS Security+: Enhancing the Data Distribution Service With TPM-based Remote Attestation. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30–August 02, 2024, Vienna, Austria.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3664476.3670442

ARES 2024, July 30–August 02, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3670442

1 INTRODUCTION

In our modern interconnected world, securely and reliably exchanging data over the network is a cornerstone of many applications. The *Data Distribution Service (DDS)* is a widely accepted industry standard maintained by the Object Management Group (OMG), which aims to provide a network reference architecture for the simple, reliable, and scalable transmission of data [18]. DDS follows a data centric publish-subscribe communication model, which allows distributed applications (also called *domain participants*) to exchange arbitrary information by publishing messages to user-defined *topics*. Applications interested in the data can receive it by subscribing to the respective topics in the global data space. The DDS middleware then takes care of transparently delivering messages from publishers to subscribers. It also handles the necessary data serialization and the underlying network programming. Application developers can configure the DDS middleware by specifying quality-of-service parameters concerning delivery methods, message reliability, and real-time constraints. Since DDS offers a fast and highly-configurable data exchange mechanism, it is currently used for a multitude of different applications, including robotics [7, 20], automotive communication [23], and sensor networks [4, 5].

In order to protect the confidentiality and integrity of critical data during transmission, the DDS standard mandates various security features. Most importantly, DDS includes a certificate-based authentication scheme for network participants, combined with a data encryption layer for transmitted messages [19]. While the DDS security model has been evaluated closely over the past few years [10, 13, 15], so far there is not much research into leveraging the advantages of *trusted computing* to enhance DDS security. For example, hardware secure elements such as *Trusted Platform Modules (TPMs)* could be used to protect DDS private keys against software-based attackers. An even more interesting feature of TPMs, which has recently gained traction, is *remote attestation*. Remote attestation denotes the approach of cryptographically verifying the integrity of a software stack that is being executed on a trusted platform. This allows applications to detect the malicious manipulation of a network peer's code base (for example due to a malware attack) before transmitting any critical information or trusting the provided services. Especially in security-critical distributed applications, providing support for transparently verifying the code integrity of individual DDS participants would greatly enhance the security and resilience of the overall system. However, to our knowledge no solution for conducting TPM-based remote attestations in DDS infrastructures has been developed so far.

In this paper, we present our proposal for integrating TPM-based remote attestation into the DDS security architecture. More concretely, this work makes the following research contributions.

(1) We develop an extension of the DDS security architecture that integrates TPM-based attestation evidence into the DDS secure channel handshake. Our proposal cryptographically binds the established secure channels to the attested software stacks, while simultaneously preserving full backwards compatibility with the DDS standard.
(2) We show how reference integrity measurements can be specified and disseminated using the DDS access control plugin.
(3) We validate our solution with a formal security analysis using the Tamarin theorem prover.
(4) We implement our proposal in the eProsima FastDDS framework and conduct a performance evaluation.

We begin this paper in section 2 with a summary of relevant previous work. Section 3 introduces background information regarding the DDS security architecture and TPM-based remote attestation protocols. In section 4, we first motivate a list of requirements for the integration of remote attestation into DDS. Based on the identified requirements, we then present in detail our proposed extension of the DDS secure channel handshake. Sections 5 and 6 contribute the security analysis of our proposal, as well as a performance evaluation of our reference implementation. We conclude this paper in section 7 with a brief summary and an outlook on future work.

2 RELATED WORK

The security of DDS-based communication infrastructures has been explored in several previous publications. Friesen et al. conduct a conceptual comparison of the DDS security architecture with classical TLS- and DTLS-based solutions [10]. Kim et al. evaluate the performance impact of using DDS security features in various configurations compared to VPNs [13]. Deng et al. follow a model-checking approach to analyze the security of the Robot Operating System (ROS2), which relies on DDS as underlying communication framework [8]. Recently, Maggi et al. conducted a comprehensive security analysis of the real-time publish-subscribe (RTPS) transport protocol, which DDS uses for message delivery [15]. Using a fuzzing approach, the authors found several vulnerabilities in six different DDS implementations and proposed corresponding mitigations.

Securing the integrity of computing systems by means of TPM-based remote attestation has been proposed for a wide variety of applications, including cloud environments [22] and the IoT [25]. However, leveraging trusted computing technologies for the Data Distribution Service is not yet well researched. While RTI Connext published a white paper outlining the use of TPM 2.0 as a hardware-backed key store in their DDS implementation [21], this proposal does not take advantage of the TPM's remote attestation capabilities. Indeed, to our knowledge the feasibility of integrating TPM-based remote attestation into DDS infrastructures has not been explored so far. Relating to this goal, however, there has been previous work regarding the support of Trusted Execution Environments (TEEs) in ROS. Mazzeo and Staffa describe how Intel's Software Guard Extensions (SGX) can be used to protect the execution of legacy ROS nodes [16]. Similarly, Beck et al. develop a ROS2-based framework for privacy-compliant drones, which uses ARM TrustZone as underlying TEE [3]. Nevertheless, neither of these proposals integrate TPM-based remote attestation into DDS.

3 BACKGROUND

In this section, we briefly introduce the security architecture of DDS, as well as the concept behind TPM-based remote attestation.

3.1 DDS Security Architecture

The DDS specification defines its security features by means of several Service Plugin Interfaces (SPIs), which any DDS middleware must implement against to provide data security for distributed applications. In total, the DDS security architecture consists of three mandatory plugin interfaces (*authentication*, *access control*, and *cryptography*), as well as two optional plugins (*data tagging* and *logging*) [19]. DDS users can independently activate and configure these security plugins for their applications, depending on the individual security needs. Hence, this plugin-based design increases flexibility and optimizes resource utilization by allowing applications to enable only the required security features. Furthermore, DDS users can also replace the default implementation of the security plugins with customized versions. Figure 1 gives an overview of the resulting DDS security model and the dependencies between the different plugins.

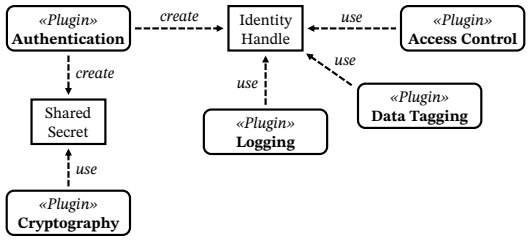

Figure 1: The DDS security model. Simplified from [19].

At the center of the DDS security architecture lies the *authentication plugin*, which uses digital certificates signed by a globally trusted *identity CA* to mutually authenticate DDS participants during channel establishment. The default authentication plugin is called `DDS:Auth:PKI-DH` and supports X.509 certificates using either RSA or ECDSA key pairs [19]. The identity details of an authenticated DDS participant (most importantly its certified subject name) is provided to other plugins in form of an *identity handle*. Furthermore, the authentication plugin also conducts an ephemeral (Elliptic Curve) Diffie-Hellman key exchange to establish a *shared secret* with the authenticated DDS participants.

The *access control* plugin reads and enforces permission policies that are disseminated by DDS participants together with their certificates. These policies must be digitally signed by a dedicated *permission CA* in order to be trustworthy. System administrators can use permission policies to selectively control the communication capabilities of individual applications within the distributed system, for example by restricting read and/or write access for topics that are carrying sensitive data. The default access control plugin is called `DDS:Access:Permissions`. It supports an XML-based policy format and uses the remote participant's identity handle, which

has been generated during authentication, to identify the active access rules for this participant.

The *cryptography plugin* is responsible for providing a transport encryption layer in DDS. The default plugin implementation derives a symmetric session key from the shared Diffie-Hellman secret that has been established during the participant authentication process [19]. It then uses AES in Galois Counter Mode (GCM) to transparently encrypt all messages exchanged with the remote participant. Finally, the *data tagging* plugin allows the addition of sticky labels to transmitted data, while the *logging* plugin provides a dedicated log environment for the security plugins. However, both of these plugins are optional and of no further relevance for the contributions presented in this paper.

3.2 Trusted Platform Modules

Trusted Platform Modules (TPMs) are cryptographic co-processors that extend computing devices with trustworthy, hardware-based security features [26]. While TPMs are almost ubiquitously available in modern server and desktop systems, in the field of embedded devices and cyber-physical systems they became prominent only recently. Similar in nature to smart cards, TPMs provide security features such as management of cryptographic keys, symmetric and asymmetric encryption, as well as digital signatures. By implementing such mechanisms in a separate hardware module, TPMs can isolate security-critical functions from the rest of the system. As a result, private parts of TPM-generated keys are usable only by the trusted hardware itself and are never released in plain text.

Platform Integrity Measurements. In addition to being useful as cryptographic co-processors, TPMs also allow to safeguard the software integrity of trusted platforms. This requires a *trusted boot process*, which consecutively logs (i.e., *measures*) a hash digest of all boot stages that are loaded and executed on the device, including the operating system kernel. Any malicious device modification or malware infection will hence be reflected by a changed digest in the measurement log. Kernel modules such as the Linux *Integrity Measurement Architecture (IMA)* can extend the chain of measurements to dynamically loaded user applications as well [12]. However, the integrity of the collected measurement log itself must also be protected against tampering on the (potentially compromised) platform. For this, the TPM offers a special region of volatile memory called *Platform Configuration Registers (PCRs)*. Each PCR stores a set of measurement digests in the form of an extensible hash chain [1]. Since the TPM only allows the *addition* of new measurements to a PCR hash chain, even attackers with privileged access to the device cannot hide the digests of compromised boot stages or user applications after they have been measured.

Remote Attestation. The process of verifying integrity measurements of a trusted platform over the network is called remote attestation. Attestations are usually conducted in form of a challenge-response protocol [24]. For this, a *relying party* generates a random challenge and sends it to the *attester* (i.e., the trusted platform). The attester then uses its TPM to generate an attestation report called a *quote*. This quote contains the received challenge for freshness, as well as the current PCR values that are stored inside the TPM. Since the PCR values are aggregated digests of all previously conducted integrity measurements, the resulting quote unambiguously represents the current software state of the attester's trusted platform [1]. To prevent any tampering of the reported PCR values, the quote is cryptographically signed by the TPM. After receiving the quote and validating its signature, the relying party can establish the attester's trustworthiness by checking the PCR digests contained in the quote against a list of expected reference values.

4 REMOTE ATTESTATION IN DDS

In this section, we present our proposal for extending the DDS authentication protocol with TPM-based remote attestation. For this, we first identify five functional requirements that shall guide our protocol design. We also show how an enhanced version of the DDS access control plugin can be used to validate the attested integrity measurements against trustworthy reference values.

4.1 Requirements

We define five requirements R1 to R5 that our solution for integrating remote attestation into DDS must fulfill.

(R1) Code base attestation: Our extension of the DDS security architecture must introduce support for the TPM-based attestation of remote participants' code bases. The attestation process should be transparently integrated into the existing DDS authentication protocol. To allow flexible configurations for different use cases, both uni- and bidirectional (i.e., mutual) remote attestation must be supported.

(R2) Integrity verification: DDS participants must be able to specify "golden" reference values for the expected PCR digests of their trustworthy software stacks. The specified reference values should be used automatically by all peers for the verification of transmitted attestation evidence.

(R3) Attested channels: The protocol extension must cryptographically bind the established secure communication channels to the attested trusted platforms. This is necessary to prevent masquerading attacks [24] and to ensure that transmitted messages can only be encrypted by trustworthy DDS participants with verified code bases.

(R4) Replay protection: Attackers must not be able to replay previously intercepted attestation evidence to impersonate a trusted platform.

(R5) Backwards compatibility: Since DDS is deployed in distributed systems, we cannot assume all participants to have a platform TPM or to run an extended DDS implementation with support for remote attestation. To prevent the partitioning of the DDS network and ensure seamless communication, our protocol extension must provide full backwards compatibility with the standard non-TPM authentication process.

4.2 Extending DDS Authentication

Our goal in this work is to extend the DDS authentication plugin with a TPM-based remote attestation process. To achieve this, we need to solve two principal design challenges.

(1) Identify a remote attestation model that is suitable for inclusion into the DDS architecture.
(2) Define a concrete attestation process that binds the secure communication channels to the attested platforms.

In the remainder of this section, we discuss how we solve these two design issues, before presenting our proposed extension of the DDS authentication handshake.

Selecting a Suitable Attestation Model. As introduced in section 3, remote attestation protocols exchange TPM-signed *quotes* that reflect a trusted platform's software state by means of the measured PCR values. However, TPM quotes do not directly contain the attested PCR values themselves. Instead, they include only a single *composite digest*, which is generated by the TPM as the hashed concatenation over all relevant PCR registers [27]. One principal concern when designing a remote attestation protocol is how to choose the PCR banks and indices that should be quoted as part of the composite digest. Most existing attestation protocols let the relying party define a suitable PCR selection during the challenge-response handshake, notably the *CHARRA* reference implementation for TPM 2.0 [6, 9]. Figure 2 illustrates the resulting remote attestation model. With this approach, the relying party must initially load the

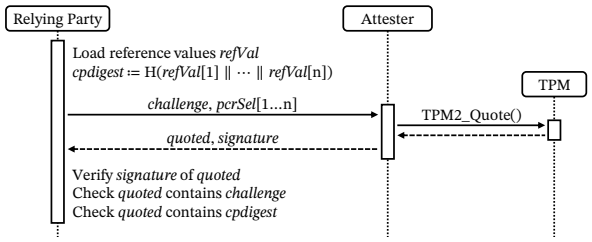

Figure 2: Basic attestation with relying-party-chosen PCRs.

set of expected PCR reference values and transmit a corresponding PCR selection (i.e., banks and register indices) to the attester. The relying party also calculates the expected PCR composite digest by concatenating and hashing the loaded reference values. On the attester's side, the TPM generates and signs a fresh quote for the requested PCR selection. Finally, the trustworthiness of the attester can be verified by checking the quote's signature and comparing the quoted PCR composite digest with the expected one.

Even though this attestation model is popular and requires only a single TPM command per request, we find it to be infeasible for integration into the DDS security architecture. This is because it necessitates the relying party to determine the PCR selection *in advance* of the attestation process. However, in order to look up the expected reference values and retrieve the correct PCR selection to request, the relying party must know the identity of the attester. Since the identity of remote DDS participants (i.e., their certificate) is established during authentication, including attestation into the DDS authentication handshake would create a circular dependency.

To still achieve a practical remote attestation mechanism in DDS, we propose an *attester-chosen* PCR selection approach instead. With this model, the attesters themselves are responsible for selecting adequate PCR registers to quote. Usually, the chosen PCR selection depends on the capabilities of the attester's TPM (e.g., the available banks) as well as the local platform configuration (e.g., the individual PCR registers in use). As figure 3 shows, this approach avoids the need for the relying party to identify expected PCR reference values and pre-compute their composite digest before the actual

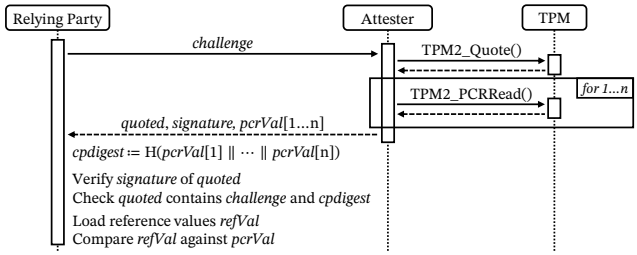

Figure 3: Basic attestation with attester-chosen PCRs.

authentication process starts. On the other hand, the attester is now forced to explicitly read and transmit the quoted PCR values for verification, since they are not directly contained in the quote. The relying party validates the authenticity of the received PCR values by calculating their composite digest and comparing it with the quoted value. The expected reference values can then be loaded and verified against the received PCR list at a later time, for example as part of the DDS access control process.

Establishing Trusted Channels. The second design challenge consists of properly binding the established secure communication channels to the attested trusted platforms. Since remote attestation protocols have been extensively researched in the past, there are already solutions for this. Stumpf et al. propose to conduct a Diffie-Hellman (DH) key exchange between attester and relying party that is authenticated via the TPM quote [24]. This can be achieved by combining a hash of the generated DH public keys with the received challenge, and then using the result as qualifying data for the `TPM2_Quote` command [24]. Similarly, established TLS channels can also be bound to trusted platforms by hashing TLS certificates into the quote [2, 31]. Mainly due to its simplicity and efficiency, this solution has since become a popular pattern to achieve secure communication channels in remote attestation protocols. However, this approach has also been identified as susceptible for nonce-data attacks by malicious platform owners under certain conditions [11]. Alternatively, it is also possible to authenticate DH public keys and TLS certificates using the PCR registers instead of the quote's qualifying data field. For this, the key material that established the secure channel must be extended into a resettable PCR register (e.g., number 16 or 23) before generating the quote. While this also cryptographically binds the secure channels to the attested platform, it requires the system-wide synchronization of remote attestation requests to prevent "overwriting" the used PCR register before the channel is fully authenticated. Finally, conducting a Diffie-Hellman key exchange using the TPM hardware itself has also been proposed [30]. The main downside of this approach is the increased complexity and a substantial performance impact, as TPMs are slow in performing asymmetric cryptography.

Since it is currently the most popular solution and also fits well with the existing DDS security architecture, for our proposal we follow the approach by Stumpf et al. [24] and authenticate the Diffie-Hellman key exchange conducted by the DDS authentication plugin via the generated TPM quotes.

Our Protocol Proposal. Based on the discussed design considerations, in the remainder of this section we present our proposal for

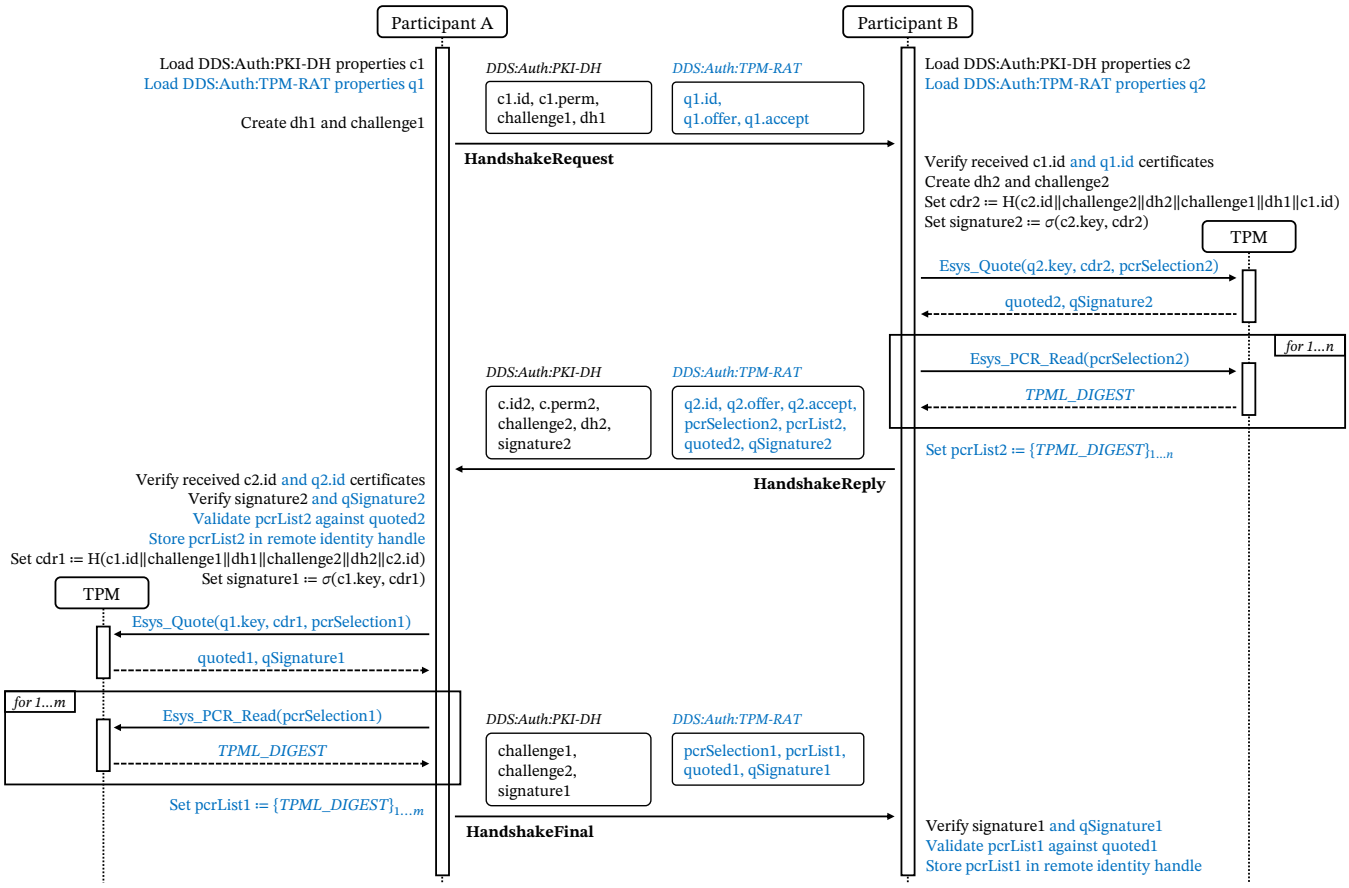

Figure 4: Our proposed extension of the `DDS:Auth:PKI-DH` authentication handshake including mutual remote attestation.

integrating a TPM-based remote attestation process into the existing DDS authentication plugin. To maximize the interoperability of our solution (c.f., requirement R5), we closely follow the standardized protocol sequence as described in [19], but enhance the default certificate-based authentication of DDS participants with additional TPM-based attestation evidence. Figure 4 shows our proposed protocol handshake between two DDS participants. The introduced attestation-specific steps and messages are displayed with blue text. First, the authentication plugin is initialized by loading the DDS quality-of-service parameters that have been configured. For certificate-based authentication, the loaded properties include a private identity key, a corresponding certificate, and an identity CA that is used for certificate validation [19]. To support remote attestation, we additionally load a TPM-generated attestation key (usually identified by a TPM handle), the certificate of the privacy CA that is used to authenticate attestation keys, and the certificate of the attestation key signed by the privacy CA. To keep the two configurations separate, we introduce the `DDS:Auth:TPM-RAT` namespace for attestation-related properties. A complete list of all new properties in this namespace is given later in section 6.

The original `DDS:Auth:PKI-DH` plugin executes a three-message handshake to mutually authenticate the two participants based on their identity certificates [19]. For this, each participant generates a new Diffie-Hellman key pair and draws a random 256-bit nonce as challenge. During the initial *HandshakeRequest* and subsequent *HandshakeResponse* messages, both participants' DH public keys, the challenges, the identity certificates (c.id), and the configured permission files (c.perm) are exchanged. To set up a remote attestation process, we extend these messages with the participants' attestation key certificates (q.id) and two boolean flags that control the quote generation (q.offer, q.accept). The trustworthiness of both received certificates is ensured by validating them against the configured identity and privacy CAs. In addition, the *HandshakeResponse* and *HandshakeFinal* messages must include a digital signature that serves as proof-of-knowledge of the participants' identity private keys. As specified in greater detail in the DDS security architecture [19], these signatures are generated over a so-called *CDR buffer*. This buffer includes (i) the transmitted challenges, (ii) the DH public keys, and (iii) the identity certificates of *both* endpoints. As a result, signing the CDR buffer (i) ensures the freshness of the exchanged signatures, (ii) binds the authentication to the conducted Diffie-Hellman key exchange, and (iii) prevents identity misbinding attacks [14]. After both participants verified the received signature and its contents, the established Diffie-Hellman shared secret is saved for later use by the DDS cryptography plugin.

As figure 4 shows, we integrate remote attestation into the DDS authentication protocol by creating, exchanging, and verifying TPM-based quotes for both participants. The quotes are generated using the `Esys_Quote` function provided by the Extended System API of the official TPM 2.0 Trusted Software Stack [29]. To protect the transmitted quotes against replay attacks (c.f., requirement R4) and bind the secure communication channels to the attested platforms (c.f., requirement R3), we use the previously assembled CDR buffers as qualifying data for the `Esys_Quote` function. Analogous to the certificate-based authentication process, this puts a hash digest of the Diffie-Hellman public keys and both random challenges into the signed quote data structure. It also binds the conducted TPM attestation to the DDS identity certificates. To transmit the resulting quotes and their signatures as part of a DDS handshake, we extend the *HandshakeResponse* and *HandshakeFinal* messages with the two additional binary fields *quoted* and *qSignature*. Following the attestation model with attester-chosen PCR selection, the list of quotable PCR banks and indices is taken directly from the local plugin configuration. Furthermore, we use the `Esys_PCR_Read` function to assemble a separate list of the current PCR values, which is also transmitted as part of the DDS protocol messages. We achieve the binary serialization of these TPM-specific message fields by leveraging the TPM 2.0 Marshalling API [28]. Note that all TPM-specific fields are seen as optional and may not always be set in the transmitted handshake messages. To validate the exchanged quotes, each participant first checks the correctness of the received quote signature against the presented attestation key certificate. This also includes verifying the quote's qualifying data, which must match the expected CDR buffer when re-calculating it using the locally stored values for the random challenges and the DH public keys. Furthermore, the received list of PCR values must be authenticated by calculating the corresponding composite digest and comparing it with the value stored inside the quote. Finally, each participant saves the list of attested PCR values in the remote identity handle for later use by the extended DDS access control plugin.

Supporting Unidirectional Attestations. Figure 4 shows our proposal for realizing a mutual remote attestation process between two DDS participants. However, we need to allow *unidirectional* attestations in DDS as well (c.f., requirement R1). This is because not every device in the DDS network necessarily has a TPM and can generate quotes. Furthermore, the attestation process causes a performance overhead (see section 6) and may not always be necessary. In our proposal, we achieve support for unidirectional remote attestation using the two boolean control flags *q.accept* and *q.offer*, which are transmitted in the *HandshakeRequest* and *HandshakeResponse* messages. These flags allow DDS participants to specify whether they are configured to provide TPM-based attestation evidence (q.offer) and whether they are able to verify received quotes (q.accept). If a DDS participant is not set up to generate quotes (e.g., because no suitable attestation key is configured), we set the q.offer flag to `false` and simply omit the TPM-specific procedures and message fields. The remote peer will then automatically skip the quote verification steps, leading to an empty set of authenticated PCR values in the remote identity handle. Similarly, if a DDS participant cannot verify quotes (e.g., because no privacy CA is configured), the q.accept flag will be set to `false`, causing the remote peer to skip the quote generation. This design allows us to flexibly support both mutual and unidirectional attestations, depending on the capabilities of the communicating DDS participants. Note that these flags do not need to be authenticated, as they only facilitate compatibility between different DDS stacks and are not security critical. We discuss this further in our security analysis in section 5.

Achieving Backwards Compatibility. In addition to supporting unidirectional attestations, we also need to ensure that our solution remains fully backwards-compatible with the original DDS authentication process (c.f., requirement R5). This means that a DDS participant executing the extended authentication handshake, as shown in fig. 4, must also be able to authenticate a remote peer that operates any other (unmodified) DDS implementation, and vice-versa. We achieve this level of compatibility by taking care not to modify the existing certificate-based authentication process at all. For example, instead of defining new protocol messages for the attestation part of the authentication handshake, we only introduce additional binary fields to the existing three-message handshake. If the contacted DDS participant does not implement support for remote attestation, these fields will be simply ignored. Furthermore, we introduce the *q.accept* flag to better control the generation and transmission of quotes in such scenarios. Any network participant with an unmodified DDS stack will not set this flag during the authentication handshake, which in turn causes the remote peer to skip the attestation process. This prevents the unnecessary generation and transmission of quotes to DDS participants that are incapable of validating them anyway.

4.3 Extending DDS Access Control

Exchanging and validating signed quotes during the DDS authentication handshake cryptographically ascertains the current PCR values (and by extension the code bases) of remote DDS participants. However, this process alone does not yet ensure that the remotely attested software stacks are indeed *trustworthy*. To achieve this, we must provide a mechanism that can automatically check the quoted PCR values of remote DDS participants against a set of expected reference values representing a good platform state. Furthermore, we must define how these PCR reference values should be specified, disseminated, and protected against tampering by malicious actors in a DDS network. Fortunately, we can accomplish these goals rather easily by building on the existing DDS access control plugin. This plugin is responsible for imposing restrictions on the behavior of authenticated DDS participants. The default access control plugin, which all DDS implementations must support, is called `DDS:Access:Permissions` [19]. It uses an XML-based policy format to represent individual *access permissions* for a set of DDS participants. Participants can be granted permission for creating, reading, and writing certain DDS topics, as well as for accessing certain DDS domains. Access permissions are specified by means of *allow rules* and *deny rules*, both of which can match either on DDS topics or domains. A set of rules is then wrapped in a *permission grant*, which is matched to a specific DDS participant using the subject name of the issued identity certificates. The resulting policies are signed by a permission CA to ensure their authenticity, before being automatically loaded and exchanged during the DDS authentication handshake (c.f., fig. 4). After a successful authentication,

the DDS implementation queries the access control plugin for each action requested by a participant, such as accessing a particular data reader or writer in the network. The requested action is only permitted if a matching grant for the implicated DDS participant is found in the remote peer's permission file.

To reach our goal of defining a practical remote attestation process for DDS, we extend the default access control plugin to (i) securely disseminate PCR reference values and (ii) transparently check the quoted digests against them. For this, we extend the plugin's XML permission schema with a new (but optional) <platform_measurements> tag. This tag allows to specify one or more sets of trustworthy PCR reference values for a particular DDS participant, which will be verified by the extended access control plugin as part of the remote attestation process. Analogous to the permission grants, we match the expected platform measurements to the received quotes using the subject name of the corresponding attestation key certificate. Each acceptable software configuration of a DDS participant is then represented by a single <pcr_selection> tag. A PCR selection includes an attribute identifying the used hash algorithm (i.e., the PCR bank) and the set of expected reference values. Listing 1 gives an example DDS permission file that specifies reference values for a trustworthy software configuration using PCR registers 0, 7, and 10. Multiple trustworthy configurations for a single DDS participant can be specified by repeating the <pcr_selection> tag. Similarly, participants using more than one PCR bank can be represented by using multiple selections with different bank attributes. The PCR values themselves are specified as colon-separated tuples of register index and a digest in hexadecimal notation. This follows the PCR output format used by the popular tpm2-tools[1] library, which simplifies the inclusion of PCR reference values into existing permission files via copy-and-paste.

Listing 1: DDS permission file with PCR reference values.

```
<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="permissionsWithPcrExtension.xsd">
    <permissions>
        <grant name="ExamplePermissions">
            <subject_name>CN=ExampleParticipant</subject_name>
            <!-- Expected platform measurements (optional) -->
            <platform_measurements>
                <!-- Identify platform via attestation key name -->
                <subject_name>CN=ExampleParticipant</subject_name>
                <!-- List of trustworthy software configurations -->
                <pcr_selection bank="sha1">
                  0  : 0xb9d349bbe15145e51db8950a6be0a590fe4fb176
                  7  : 0x4a145531060319557d4cfdc11feca52c01ef6db9
                  10 : 0x0fda683cac3484d1e7cebc615e05f67c52260cd4
                </pcr_selection>
            </platform_measurements>
            <allow_rule>
                <subscribe>
                    <topics>
                        <topic>ExampleTopic</topic>
                    </topics>
                </subscribe>
            </allow_rule>
            <default>DENY</default>
        </grant>
    </permissions>
</dds>
```

Our proposed extension of the DDS permission schema allows system administrators to specify the expected platform measurements for their systems directly via access control policies. These policies are then automatically disseminated by the DDS middleware during the authentication handshake (c.f., fig. 4). Since valid DDS permission files must always be signed by a globally trusted permission CA, the integrity of the included PCR reference values is secured against malicious tampering. The transmitted permission files are then evaluated and enforced by the remote peer's access control plugin. To facilitate this, our extended access control plugin parses the received policies according to the updated XML schema and extracts the specified platform measurements for each permission grant. Every time a DDS network operation is requested, the access control plugin searches the received policy for a permission that matches the authenticated remote participant [19]. We augment this permission finding process by additionally looking for a matching PCR selection in the iterated grants. A PCR selection matches if (i) the subject name of the used attestation key certificate matches and (ii) *all* expected PCR registers of this selection are quoted and their digests match. If no matching permission grant is found in the policy, the access control plugin will deny the communication with the remote DDS participant. However, note that specifying trustworthy platform measurements for a DDS participant is optional in our permission schema (c.f., listing 1). Hence, if a permission grant does not demand any specific platform measurements, it will match regardless of the remote participant's attested PCR values. This design ensures that the access control plugin can also handle unidirectional attestations (c.f., requirement R1).

Finally, we must also ensure the backwards compatibility of our modifications (c.f., requirement R5). Here the main issue is that unmodified DDS stacks must still be able to parse and evaluate our extended access control policies, even if they include PCR reference values for certain participants. To ensure this, we define a separate XSD schema for our updated policy model, which the administrators of TPM-enabled DDS participants should reference in their policies. This allows DDS participants that do not support remote attestations to still parse the received access control policies. The included PCR reference values are then simply ignored.

5 SECURITY ANALYSIS

In this section, we determine the security of our solution by informally showing that the requirements R1 to R5 are fulfilled. Furthermore, we present our approach for a formal verification of the proposed DDS authentication protocol using the Tamarin prover.

5.1 Threat Model

We consider malicious DDS participants to be the main adversaries in our scenario. This includes (i) attacker-controlled DDS nodes attempting to join the network, as well as (ii) legitimate nodes that have been compromised by malware. In both cases, DDS nodes with untrustworthy code bases must be excluded from communicating with the rest of the distributed system. Furthermore, the communication between attested DDS participants must be protected against evesdropping by general network attackers. Finally, we assume the used identity, permission, and privacy CAs to be trustworthy.

5.2 Informal Security Analysis

The primary goal of our proposed DDS extension is to provide support for the transparent cryptographic verification of remote participants' code bases (requirement R1). We achieve this by automatically generating and exchanging TPM-based quotes as part of

[1] https://tpm2-tools.readthedocs.io

the DDS authentication handshake (c.f., fig. 4). These quotes unambiguously represent the underlying trusted software stacks of the involved DDS participants and are digitally signed by TPM-internal attestation keys to prevent forgery or tampering. We introduce an additional privacy CA to authenticate the attestation keys in the same manner as the existing DDS identity certificates. Furthermore, our solution is designed to provide support for *both* unidirectional and mutual remote attestations. The used attestation mode is dynamically selected depending on the capabilities and configuration of the two communicating participants. We seamlessly integrate the proposed attestation mechanism into the existing DDS security architecture by piggybacking on the existing three-message security handshake. This also ensures that our protocol extension does not influence the certificate-based DDS authentication process.

The second important requirement concerns the validation of the attested code bases' trustworthiness (requirement R2). We achieve this goal by extending the DDS access control policy schema to specify PCR reference values for a set of DDS participants. The augmented policies are then automatically exchanged, verified, and enforced at the relying parties. As a result, communicating with remotely attested DDS participants is permitted only if the attested PCR values are authorized by the access control policies. We ensure the integrity of the specified reference values using the existing DDS permissions CAs. Note that our proposal of utilizing the DDS access control plugin to automatically validate attested PCR values also protects the remote attestation mechanism against downgrade attacks. Because our solution needs to support both mutual and unidirectional attestations, quotes are considered to be optional fields in the extended DDS authentication handshake (c.f., fig. 4). Hence, a network attacker could downgrade a bidirectional to a unidirectional attestation by removing quotes from the transmitted messages. Our extended DDS access control plugin prevents such attacks, since the resulting empty set of attested PCR values will not be authorized by the participant's access control policy.

Besides verifying the integrity of remote code bases, our solution also needs to establish secure communication channels between the attested DDS participants (requirement R3). For this, we bind the encrypted DDS communication channels to the attested platforms by including a digest of both Diffie-Hellman public keys into the signed quote. This authenticates the conducted key exchange with the TPM-internal attestation keys, which are protected by the trusted hardware. Since attackers cannot forge signatures for self-chosen quotes, man-in-the-middle attacks on the established communication channels are thus effectively prevented. This is true even if the used DDS identity certificates are compromised, because our remote attestation mechanism is completely independent of the existing DDS authentication procedure. Note that we only propose to add a second channel authentication to the existing protocol handshake, but do not modify the underlying security mechanisms. Hence, we still rely on the default DDS authentication plugin to implement the Diffie-Hellman key exchange and on the DDS cryptography plugin to establish the symmetric encryption layer. This reduces the complexity of our solution and ensures that we do not influence or weaken any existing DDS security features.

Considering the goal of replay protection (requirement R4), we ensure quote freshness by including randomly drawn nonces into the signed data structures. Furthermore, we achieve backwards compatibility (requirement R5) by designing our DDS extension such that all changes remain transparent to unmodified DDS participants. For instance, our extended authentication handshake only adds optional fields to existing protocol messages, which are ignored by unmodified DDS software stacks. Similarly, our enhanced access control policies can be parsed even by non-TPM participants using the referenced XSD schema definition.

5.3 Formal Security Analysis

To verify the security of our proposed DDS authentication protocol extension, we modeled the handshake shown in fig. 4 with the Tamarin theorem prover. Tamarin is a tool to formally verify cryptographic protocols and their security properties using symbolic analysis of first-order logic terms [17]. We modeled the three protocol messages defined in fig. 4 as Tamarin rules, leveraging the built-in equational theories for hashing, Diffie-Hellman key exchanges, and digital signatures. The protocol's security properties are then formalized as first-order logic terms quantifying over time points [17]. In our case, successfully executing the protocol handshake must imply that both endpoints are attested and have established a shared secret, but no (active) network adversary may have learned it. The resulting formal model of our proposed DDS remote attestation protocol is given in appendix A.

6 IMPLEMENTATION AND EVALUATION

We implemented and tested our proposed extension of the DDS security architecture as a fork of the popular eProsima FastDDS middleware[2]. Our proof-of-concept implementation includes an enhanced DDS authentication plugin that transparently conducts TPM-based remote attestations between DDS participants as described in section 4. Table 1 shows the additionally introduced quality-of-service properties that administrators can use to configure the participants' remote attestation capabilities. If no properties are set, the participant will not offer a remote attestation endpoint.

`privacy_ca`	Root CA certificate used to verify attestation keys.
`attestation_key`	TPM-internal attestation key to use for quoting. Can be specified as TPM handle or as object file.
`attestation_cert`	CA-signed certificate of the attestation key.
`pcr_banks`	Comma-separated list of PCR banks to quote. Supported: sha1, sha256, sha384, sha512, sm3_256.
`pcr_selection`	Comma-separated range of registers to quote.
`tcti_options`	Additional options to initialize the TPM Command Transmission Interface (TCTI).

Table 1: New properties in the `DDS:Auth:TPM-RAT` namespace.

We also updated the FastDDS access control plugin to parse extended permission policies and authorize attested remote platforms based on the specified PCR reference values. However, since FastDDS implements a naive XML parser that does not support XSD schema validation, backwards compatibility of the access control policies can be achieved only by specifying the reference values at the *end* of the `<grant>` tag. Our complete proof-of-concept, including several usage examples, is available under the Apache 2.0 license[3].

[2] https://github.com/eProsima/Fast-DDS
[3] https://gitlab.cc-asp.fraunhofer.de/ros2-security/fast-dds-tpm

Performance Evaluation. We demonstrate the practicality of our proposal for integrating TPM-based remote attestation into DDS by means of a performance evaluation. For this, we measured the mean connection time of DDS participants using (i) no security, (ii) the default certificate-based authentication handshake, and (iii) both the unidirectional as well as mutual attestation procedures as presented in section 4. To determine the scalability of TPM-based remote attestation in DDS, we also tested the establishment of up to eight parallel connections between individual DDS participants. Our evaluation platform consists of a Thinkpad T480s (Ubuntu 22.04 LTS, 16 GB memory) with an Infineon SLB9670 TPM 2.0 hardware module. The used benchmark implementation is available online for reproducibility purposes[4].

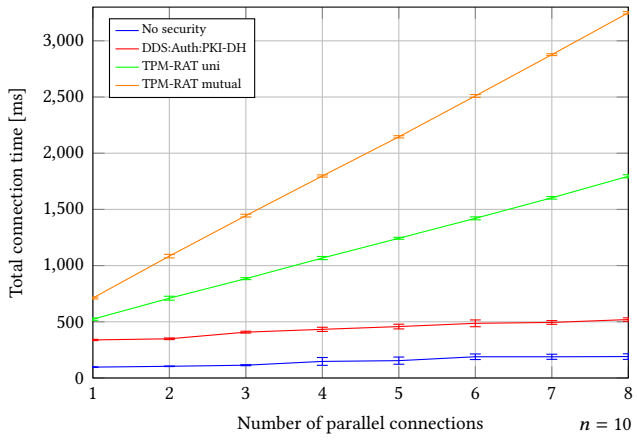

Figure 5: Connection times of DDS participants.

Our results in fig. 5 show that establishing a single DDS connection takes about 100ms without security, and about 340ms using certificate-based authentication. Conducting a TPM-based remote attestation on top of that increases the mean connection time to about 520ms unidirectionally and 710ms bidirectionally. This overhead is caused almost exclusively by the time required to create and sign quotes on the secure hardware. Furthermore, our evaluation also shows an impact on the scalability compared to standard DDS security. While the default DDS authentication only takes about 520ms to execute 8 handshakes in parallel, with TPMs this time increases to about 1.8 and 3.2 seconds, respectively. The reason for this is that the TPM, as a dedicated hardware resource, cannot be parallelized. However, note that our proposal only affects the *initial connection time* between two DDS participants. The data transport layer, and hence the latency and throughput of messages transmitted during system operation, is not impacted by this.

7 CONCLUSION

In this work, we present and evaluate a solution to integrate TPM-based remote attestation into the Data Distribution Service (DDS). We achieve this by extending the DDS authentication handshake with TPM-based attestation evidence. While generic attestation frameworks such as Keylime periodically scan the network to detect compromised software stacks [22], our solution instead focuses on transparently establishing secure and attested communication channels *between* DDS participants. In addition, we propose an enhanced DDS access control schema that can disseminate and validate reference integrity measurements directly in the existing DDS ecosystem. Our solution is transparent to DDS participants and preserves full backwards compatibility with the DDS standard. We validate our proposal by means of a security analysis including the formal verification of our protocol, as well as a performance evaluation using a popular hardware TPM.

As future work, we plan to integrate and evaluate our solution in other DDS-based communication frameworks as well, most importantly ROS2. For the contributions presented in this paper, we simply assume TPM-based attestation keys to be available and properly authenticated on all DDS nodes. However, introducing remote attestation to high-level frameworks such as ROS2 also necessitates adequate tool support for administrators to properly provision TPM-based attestation keys and handle certificate revocation. In addition, since our evaluation in section 6 revealed a noticeable performance impact, mitigating denial-of-service attacks on attested DDS infrastructures constitutes an important topic of future research as well. Finally, we also plan to investigate the possibilities of other trusted computing technologies, for example ARM TrustZone, to enhance the security of DDS infrastructures even further.

[4]https://gitlab.cc-asp.fraunhofer.de/ros2-security/fast-dds-tpm/-/tree/2.13.2-tpm/examples/cpp/dds/PerformanceTestWithTPMExample

ACKNOWLEDGMENTS

This work was funded by the Helmholtz Association (HGF) through the Competence Center for Applied Security Technology (KASTEL), subtopic 46.23.04 Engineering Security for Production Systems.

REFERENCES

[1] Will Arthur, David Challener, and Kenneth Goldman. 2015. *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security.* Springer Nature.
[2] NorazahAbd Aziz, Nur Izura Udzir, and Ramlan Mahmod. 2014. Extending TLS with Mutual Attestation for Platform Integrity Assurance. 9, 1 (2014), 63–72.
[3] Rakesh Rajan Beck, Abhishek Vijeev, and Vinod Ganapathy. 2020. Privaros: A framework for privacy-compliant delivery drones. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security.* 181–194.
[4] Kai Beckmann and Olga Dedi. 2015. sDDS: A portable data distribution service implementation for WSN and IoT platforms. In *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES).* IEEE, 115–120.
[5] Kai Beckmann and Marcus Thoss. 2012. A wireless sensor network protocol for the OMG data distribution service. In *Proceedings of the 10th International Workshop on Intelligent Solutions in Embedded Systems.* IEEE, 45–50.
[6] Henk Birkholz, Dave Thaler, Michael Richardson, Ned Smith, and Wei Pan. 2023. Remote ATtestation procedureS (RATS) Architecture. https://datatracker.ietf.org/doc/rfc9334
[7] Jesús Martínez Cruz, Adrián Romero-Garcés, Juan Pedro Bandera Rubio, Rebeca Marfil Robles, and Antonio Bandera Rubio. 2012. A DDS-based middleware for quality-of-service and high-performance networked robotics. *Concurrency and Computation: Practice and Experience* 24, 16 (2012), 1940–1952.
[8] Gelei Deng, Guowen Xu, Yuan Zhou, Tianwei Zhang, and Yang Liu. 2022. On the (In)Security of Secure ROS2. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22).* Association for Computing Machinery, New York, NY, USA, 739–753. https://doi.org/10.1145/3548606.3560681
[9] Fraunhofer SIT. 2024. *CHARRA: CHAllenge-Response Based Remote Attestation with TPM 2.0.* https://github.com/Fraunhofer-SIT/charra
[10] Maxim Friesen, Gajasri Karthikeyan, Stefan Heiss, Lukasz Wisniewski, and Henning Trsek. 2020. 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, 201–216.

[11] Yacine Gasmi, Ahmad-Reza Sadeghi, Patrick Stewin, Martin Unger, and N Asokan. 2007. Beyond Secure Channels. In *Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing* (2007). 30–40.

[12] Gentoo Linux. 2021. Integrity Measurement Architecture. https://wiki.gentoo.org/wiki/Integrity_Measurement_Architecture

[13] Jongkil Kim, Jonathon M Smereka, Calvin Cheung, Surya Nepal, and Marthie Grobler. 2018. Security and performance considerations in ros 2: A balancing act. *arXiv preprint arXiv:1809.09566* (2018).

[14] Hugo Krawczyk. 2003. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In *Advances in Cryptology - CRYPTO 2003*, Dan Boneh (Ed.). Vol. 2729. Springer Berlin Heidelberg, 400–425. https://doi.org/10.1007/978-3-540-45146-4_24

[15] Federico Maggi, Rainer Vosseler, Mars Cheng, Patrick Kuo, Chizuru Toyama, T Yen, and E Boasson V Vilches. 2022. A Security Analysis of the Data Distribution Service (DDS) Protocol. *Trend Micro Research* (2022).

[16] Giovanni Mazzeo and Mariacarla Staffa. 2020. TROS: Protecting Humanoids ROS from Privileged Attackers. *International Journal of Social Robotics* 12, 3 (2020), 827–841.

[17] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *25th International Conference on Computer Aided Verification* (2013). Springer, 696–701.

[18] Object Management Group. 2015. OMG Data Distribution Service (DDS). https://www.omg.org/spec/DDS/1.4/PDF

[19] Object Management Group. 2018. DDS Security. https://www.omg.org/spec/DDS-SECURITY/1.1/PDF

[20] Jesús Rodríguez-Molina, Sonia Bilbao, Belén Martínez, Mirgita Frasheri, and Baran Cürüklü. 2017. An optimized, data distribution service-based solution for reliable data exchange among autonomous underwater vehicles. *Sensors* 17, 8 (2017), 1802.

[21] RTI Connext. 2019. Using TPM 2.0 with RTI Connext DDS Secure. https://d2vkrkwbbxbylk.cloudfront.net/sites/default/files/using_tpm_2.0_with_dds_secure.pdf

[22] Nabil Schear, Patrick T. Cable, Thomas M. Moyer, Bryan Richard, and Robert Rudd. 2016. Bootstrapping and Maintaining Trust in the Cloud. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications* (2016). 65–77.

[23] Claudio Scordino, Angela Gonzalez Mariño, and Francesc Fons. 2022. Hardware acceleration of data distribution service (DDS) for automotive communication and computing. *IEEE Access* 10 (2022), 109626–109651.

[24] Frederic Stumpf, Omid Tafreschi, Patrick Röder, Claudia Eckert, et al. 2006. A Robust Integrity Reporting Protocol for Remote Attestation. In *Proceedings of the Workshop on Advances in Trusted Computing (WATC)* (2006). 65.

[25] Hailun Tan, Gene Tsudik, and Sanjay Jha. 2019. MTRA: Multi-Tier randomized remote attestation in IoT networks. *Computers & Security* 81 (2019), 78–93.

[26] Trusted Computing Group. 2019. Trusted Platform Module Library Part 1: Architecture. https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf

[27] Trusted Computing Group. 2019. Trusted Platform Module Library Part 3: Commands. https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part3_Commands_pub.pdf

[28] Trusted Computing Group. 2020. TCG TSS 2.0 Marshalling/Unmarshalling API Specification. https://trustedcomputinggroup.org/wp-content/uploads/TCG_TSS_Marshaling_Unmarshaling_API_v1p0_r07_pub.pdf

[29] Trusted Computing Group. 2021. TCG TSS 2.0 Enhanced System API (ESAPI) Specification. https://trustedcomputinggroup.org/wp-content/uploads/TSS_ESAPI_v1p0_r14_pub10012021.pdf

[30] Paul Georg Wagner, Pascal Birnstill, and Jürgen Beyerer. 2020. Establishing Secure Communication Channels Using Remote Attestation with TPM 2.0. In *Security and Trust Management* (2020). 73–89.

[31] Robert Walther, Carsten Weinhold, and Michael Roitzsch. 2022. RATLS: Integrating Transport Layer Security with Remote Attestation. In *Applied Cryptography and Network Security Workshops: ACNS 2022 Satellite Workshops* (2022). 361–379.

A APPENDIX: PROTOCOL FORMALIZATION

```
theory DDSAuthTPM
begin
builtins: hashing, signing, diffie-hellman

// Define two DDS participants (Alice and Bob)
rule Participants: [] --> [Participant('A'), Participant('B')]

// Create identity keys and certificates
rule CreateIdentityKey:
  [ Fr(~key), Participant(X) ]
  --[
    OnlyOnceFor(<'CreateIdentityKey', X>), IsIdentityKey(X, ~key)
  ]->[ Id(X, ~key), !IdCert(X, pk(~key)) ]

rule IdentityCertsArePublic:
  [ !IdCert('A', idCertA), !IdCert('B', idCertB) ]
  -->[ Out(idCertA), Out(idCertB) ]

// Create attestation keys and certificates
rule CreateAttestationKey:
  [ Fr(~ak), Participant(X) ]
  --[
    OnlyOnceFor(<'CreateAttestationKey', X>), IsAk(X, ~ak)
  ]->[ Ak(X, ~ak), !AkCert(X, pk(~ak)) ]

rule AkCertsArePublic:
  [ !AkCert('A', akCertA), !AkCert('B', akCertB) ]
  -->[ Out(akCertA), Out(akCertB) ]

// Execute DDS authentication protocol between Alice and Bob
rule HandshakeRequestMessage:
  [ Fr(~nonceA)                    // Choose fresh nonce for Alice
    , Fr(~a)                       // Choose fresh DH private key for Alice
    , !IdCert('A', idCertA)        // Load Alice's identity certificate
    , !AkCert('A', akCertA)        // Load Alice's quoting certificate
  ]--[
    AuthRequest('A', 'B', ~nonceA)  // Note Alice's authentication request
  ]->
  [ // Send message to Bob:
    // c.id = idCertA, q.id = akCertA, challenge1 = nonceA, dh1 = g^a
    Out(<'HandshakeRequestMessage', 'A', idCertA, akCertA, ~nonceA, 'g'^~a>)
    , Alice_('B', ~nonceA, ~a)     // Save Alice's session
  ]

rule HandshakeReplyMessage:
// According to the DDS security specification, the signature contains:
//   hash_c2, challenge2, dh2, challenge1, dh1, and hash_c1 (in that order)
let cdrbuf = <h(idCertB), ~nonceB, 'g'^~b, challenge1, dh1, h(idCertA)>
    sigB = sign(cdrbuf, keyB)
    quoteB = sign(<cdrbuf, 'PCRB'>, akB)
in
  [ In(<'HandshakeRequestMessage', A, c_id, q_id, challenge1, dh1>)
    , Fr(~nonceB)                                  // Choose fresh nonce for Bob
    , Fr(~b)                                       // Choose fresh DH key
    , Id('B', keyB), !IdCert('B', idCertB)         // Load Bob's identity key
    , Ak('B', akB), !AkCert('B', akCertB)          // Load Bob's quoting key
    , !IdCert(A, idCertA), !AkCert(A, akCertA)     // Load Alice's certificates
  ]
  --[
    Neq(dh1, 'g')
    , Eq(idCertA, c_id), Eq(akCertA, q_id)         // Verify certificates
    , AuthResponse('B', A, ~nonceB, sigB, quoteB)  // Note Bob's response
  ]->
  [ // Send message to Alice: c.id = idCertB, q.id = akCertB, challenge1,
    // challenge2 = nonceB, dh2 = g^b, signature = sigB, qSignature = quoteB
    Out(<'HandshakeReplyMessage', 'B', idCertB, akCertB, challenge1,
                                          ~nonceB, 'g'^~b, sigB, quoteB>)
    , Bob_(A, ~nonceB, ~b, dh1)                    // Save Bob's session
  ]

rule HandshakeFinal:
// According to the DDS security specification, the signature contains:
//   hash_c1, challenge1, dh1, challenge2, dh2, and hash_c2 (in that order)
let cdrbuf = <h(idCertA), nonceA, 'g'^a, challenge2, dh2, h(idCertB)>
    sigA = sign(cdrbuf, keyA)
    quoteA = sign(<cdrbuf, 'PCRA'>, akA)
in
  [ In(<'HandshakeReplyMessage', B, c_id, q_id, challenge1, challenge2,
                                           dh2, signature, qSignature>)
    , Alice_(B, nonceA, a)                         // Load Alice's session
    , Id('A', keyA), !IdCert('A', idCertA)         // Load Alice's identity key
    , Ak('A', akA), !AkCert('A', akCertA)          // Load Alice's quoting key
    , !IdCert(B, idCertB), !AkCert(B, akCertB)     // Load Bob's certificates
  ]--[
    Neq(dh2, 'g')
    , Eq(idCertB, c_id), Eq(akCertB, q_id)         // Verify certificates
    , Eq(nonceA, challenge1)                       // Verify nonce matches
    , Eq(verify(signature,                         // Verify the signature
              <h(idCertB), challenge2, dh2, challenge1, 'g'^a, h(idCertA)>,
              idCertB), true)
    , Eq(verify(qSignature,                        // Verify the quote
              <<h(idCertB), challenge2, dh2, challenge1, 'g'^a, h(idCertA)>,
              'PCRB'>, akCertB), true)
    , AuthFinal('A', B, sigA, quoteA)              // Note Alice's auth finish
    , Attested('A', B, challenge1, akCertB)        // Note Alice attested Bob
    , EstablishedSecret('A', dh2^a)                // Note Alice's shared secret
  ]->
  [ // Send message to Bob:
    // challenge1, challenge2, signature = sigA, qSignature = quoteA
    Out(<'HandshakeFinal', 'A', challenge1, challenge2, sigA, quoteA>)
  ]

rule HandshakeFinal2:
  [ In(<'HandshakeFinal', A, challenge1, challenge2, signature, qSignature>)
```

```
      , Bob_(A, nonceB, b, dh1)                  // Load Bob's session
      , !IdCert('B', idCertB)                    // Load Bob's certificate
      , !IdCert(A, idCertA), !AkCert(A, akCertA) // Load Alice's certificates
    ]
    --[
      Eq(nonceB, challenge2)                     // Verify nonce matches
      , Eq(verify(signature,                     // Verify the signature
                  <h(idCertA), challenge1, dh1, challenge2, 'g'^b, h(idCertB)>,
                  idCertA), true)
      , Eq(verify(qSignature,                    // Verify the quote
                  <<h(idCertA), challenge1, dh1, challenge2, 'g'^b, h(idCertB)>,
                  'PCRA'>, akCertA), true)
      , Attested('B', A, challenge2, akCertA)    // Note Bob attested Alice
      , EstablishedSecret('B', dh1^b)            // Note Bob's shared secret
    ]->[ ]

restriction Equality:
  "All x y #i. Eq(x,y) @i ==> x = y"
restriction InEquality:
  "All x y #i. Neq(x,y) @i ==> not(x = y)"
restriction OnlyOnceFor:
  "All X #i #j. OnlyOnceFor(X)@#i & OnlyOnceFor(X)@#j ==> #i = #j"

/* Prove that Alice and Bob mutually authenticate each other and establish
 * a shared secret. This lemma ensures that the model is fully executed. */
lemma HonestProtocol:
  exists-trace
  Ex c1 c2 dh1 dh2 idCertA idCertB keyA keyB akA akB secret #i #j #k #l #m.
    AuthRequest('A', 'B', c1) @ #i
    & AuthResponse('B', 'A', c2,
       sign(<h(idCertB), c2, dh2, c1, dh1, h(idCertA)>, keyB),
       sign(<<h(idCertB), c2, dh2, c1, dh1, h(idCertA)>, 'PCRB'>,akB)) @ #j
    & AuthFinal('A', 'B',
       sign(<h(idCertA), c1, dh1, c2, dh2, h(idCertB)>, keyA),
       sign(<<h(idCertA), c1, dh1, c2, dh2, h(idCertB)>, 'PCRA'>, akA)) @ #k
    & EstablishedSecret('A', secret) @ #l
    & EstablishedSecret('B', secret) @ #m

/* Prove that an attacker cannot intercept the established secret. This
 * lemma is reduced to the security of an authenticated DHKE. */
lemma EstablishedKeySecrecy:
  not( /* It cannot be that */
    Ex c1 c2 sigA sigB quoteA quoteB secret #i #j #k #l #m #n.
      /* Alice and Bob authenticated each other, */
      AuthRequest('A', 'B', c1) @ #i
      & AuthResponse('B', 'A', c2, sigB, quoteB) @ #j
      & AuthFinal('A', 'B', sigA, quoteA) @ #k
      /* they established a shared secret, */
      & EstablishedSecret('A', secret) @ #l
      & EstablishedSecret('B', secret) @ #m
      /* and the adversary knows the secret */
      & K(secret) @ #n)

/* Prove that Alice and Bob mutually attest to each other during the
 * authentication, while establishing a single shared secret. */
lemma MutualAttestation:
  exists-trace
  Ex akA akB nonceA nonceB secret #i #j #k #l.
    IsAk('A', akA) @ #i & IsAk('B', akB) @ #j
    & Attested('A', 'B', nonceA, pk(akB)) @ #k
    & EstablishedSecret('A', secret) @ #k
    & Attested('B', 'A', nonceB, pk(akA)) @ #l
    & EstablishedSecret('B', secret) @ #l
```