A comparative evaluation of security mechanisms in DDS, TLS and DTLS

Maxim Friesen[1], Gajasri Karthikeyan[1], Stefan Heiss[1], Lukasz Wisniewski[1], Henning Trsek[2]

[1]inIT - Institute Industrial IT
Technische Hochschule Ostwestfalen-Lippe
32657 Lemgo
{maxim.friesen, gajasri.karthikeyan, stefan.heiss, lukasz.wisniewski}@th-owl.de

[2]rt-solutions.de GmbH
50968 Cologne
trsek@rt-solutions.de

Abstract. In this paper the end-to-end security mechanisms of the Transport Layer Security (TLS) as well as the Datagram Transport Layer Security (DTLS) standard and the security related plugins within the Data Distribution Service (DDS) specification are analyzed and compared. The basic IT security requirements with regard to industrial applications are defined. Both, TLS/DTLS and DDS Security are evaluated against these requirements, and features such as cryptographic keys, key exchange mechanisms, encryption algorithms and authentication methods are compared. The results shall indicate if and why the use of a DDS-specific security protocol is necessary instead of deploying TLS/DTLS. Furthermore, the fundamental differences between TLS and DTLS are discussed and the distinctive features of DDS Security are highlighted.

1 Introduction

The continuous advances of trends like *Industrie 4.0* (I4.0), the *Internet of Things* (IoT), and *Cyber-Physical Systems* (CPS) reshape the basic structure of communication systems in the industrial domain. The decentralization of systems and applications like remote monitoring, controlling and configuration, cloud computing, and machine-to-machine communication require remote access to communication networks and subsequently create big challenges for the current IT security methods. The development of components and systems by different manufacturers and the varying requirements for industrial applications lead to various proprietary communication protocols. The lack of interoperability between different systems is tackled by middleware protocols and frameworks, such as MQTT [OAS15], DDS [Obj14], OPC UA [MLD09], web services and many more.

Previous work in [FSK+18] showed that there is a wide variety of middleware solutions in the industrial domain. And all of these approaches have varying

J. Jasperneite, V. Lohweg (Hrsg.), *Kommunikation und Bildverarbeitung in der Automation*,
Technologien für die intelligente Automation 12, https://doi.org/10.1007/978-3-662-59895-5_15

implementations of common IT security features. While features like client authentication and authorization are usually realized within a middleware-specific IT security infrastructure, most middlewares utilize the Transport Layer Security (TLS) [DR08] or Datagram TLS (DTLS) [RM12] standards to provide a secure channel for data exchange. In contrast, the Data Distribution Service (DDS) protocol implements its own authentication and encryption layer [Obj18]. As industrial networks are becoming more vertically-oriented, where shop floor operations need to stay in direct contact with manufacturing execution systems or cloud computing that interconnects different locations, the use of multiple middlewares for different application purposes has to be expected. Different implementations of security features in each middleware requires users to maintain separate security infrastructures that can become unnecessarily difficult to manage. A joint IT security infrastructure that serves all middlewares would facilitate the setup and management of such a complex multi-middleware system and benefit the scalability of a network. The use of an alternative security approach within DDS contradicts this and potentially impedes compatibility to middlewares that make use of the TLS/DTLS standard. Hence, it is required to compare the DDS and TLS/DTLS approaches and evaluate their differences.

This paper first defines IT security requirements for industrial applications in Section 2. In Section 3 the general DDS architecture and its security features, as well as the TLS and DTLS standards are described. The approaches are compared and evaluated against the defined IT security requirements in Section 4. The main differences and intentions behind the alternative DDS security mechanisms are discussed. The paper is concluded in Section 5.

2 IT security requirements in Industrie 4.0

Based on the security objectives defined in the ISO/IEC 27001 standard, this section lists the main requirements for information security and puts them into an industrial context. These shall be used to evaluate the DDS Security and TLS/DTLS functions in the following sections.

Authenticity – of users or devices represents the main requisite to achieve most other security objectives. Before encrypted messages are decrypted, or checked for their integrity, it needs to be ensured that the originator is truly who it claims to be. In an industrial environment, access to machine data or its maintenance controls should only be granted to entities that can reliably authenticate themselves. Not only to be able to track and associate any network activity to a unique identity to provide *Non-repudiation* but also to apply *Access Control* mechanisms. User IDs and passwords are standard procedures. More sophisticated and secure approaches include certificates. They provide a description and a digital signature of an entity's identity. This certificate is additionally signed by a third party that both communicating partners trust, usually referred to as Certificate Authority (CA), to verify the certificate's contents.

Confidentiality – of information is required if sensitive digital data shall only be viewed by authorized entities. It therefore needs to be securely stored or

transported. When data is transmitted over an unsecured channel, an attacker that has gained access to the channel can eavesdrop and obtain all transmitted messages. Communication between separated factory sites usually contains confidential company-related information like trade secrets or personal data and needs protection. A common method to ensure confidentiality is data encryption. An encryption key is used to encrypt information or a message. An authorized recipient that is in possession of the key can decrypt and view the message. Different encryption algorithms use different mathematical methods to obfuscate digital information. The two main schemes are symmetrical and asymmetrical encryption. Symmetrical encryption implies the use of a single key for encrypting and decrypting while asymmetrical encryption uses a mutually generated key-pair which does not require a preceding key exchange. As symmetric-key encryption is much faster, it is generally used to encrypt the actual data, while asymmetric schemes are only used to exchange the symmetric key during initialization.

Integrity – of data involves maintaining its accuracy, consistency and trustworthiness throughout its whole life cycle. Machine communication can be intercepted and the sensor or control data can be tampered to harm or even destroy systems. But even non-human-caused events such as electromagnetic interference or software crashes can lead to wrong or faulty alterations in data. A common approach to detect forged or altered data are cryptographic checksums and hash functions that generate Message Authentication Codes (MACs).

Non-repudiation – is the assurance that the execution of previous actions cannot be denied. In the context of communication, it provides technical proof of the authorship of messages. A user with malicious intent cannot access and sabotage information or certain processes within a company and repudiate his actions afterwards. Digital signatures in messages provide indisputable proof of its originator. Based on asymmetric cryptographic schemes, a digital signature is calculated with a private key only the message author has access to. Recipients can verify the signature against a corresponding unique public key.

Availability – of resources and information ensures that authorized parties can access it when needed. The value of information is lost if it cannot be accessed at the right time. Services within a factory that can be accessed through the Internet to enable cloud-computing or for other purposes can be a potential target for distributed denial-of-service (DDoS) attacks. Further factors that could lead to unavailability of information may include hardware failures, software issues, power outages or natural disasters. Besides regular hardware maintenance and software patching, redundant systems and high-availability clusters with fail-over routines can provide enhanced availability in the event of failure.

Access Control – is the selective restriction of authenticated entities to services, data and other resources. In the case of few users and few resources, access control can be configured individually for each entity, but in large distributed systems with many users and hierarchical levels more sophisticated approaches are required.

3 Fundamentals

3.1 DDS Core

Data Distribution Service is a middleware communication standard maintained by the Object Management Group (OMG) [Obj]. In the context of distributed systems, the term middleware refers to a software layer between the application and the operating system. It abstracts various functions to simplify the connection of components in a system and allows developers and users to focus on the application functionality rather than on providing mechanics for the exchange of data. DDS is a data-centric publish-subscribe protocol that targets the data sharing needs of highly scalable computing environments. In message-centric middlewares, messages are just passed between applications. Data-centricity on the other hand implies the addition of contextual information to messages so that applications can interpret the received data more easily. This makes the middleware aware of the data it processes and allows it to adapt its data sharing to the needs of the application.

The DDS specification is separated in several parts and describes the provided services in the Unified Modeling Language (UML) based on a Platform Independent Model (PIM). The PIM ensures the portability of implementations in any programming language and on any operating system. The DDS Core specifications include the Real-Time Publish-Subscribe (RTPS) [Obj14] protocol and the Data-Centric Publish-Subscribe (DCPS) layer [Obj15].

Data-Centric Publish-Subscribe Architecture The DDS architecture is decentralized to provide high reliability and low-latency data connectivity for mission-critical IoT applications. As seen in Fig. 1, data in DDS is made available within a global data space consisting of DDS Domains. Individual appli-

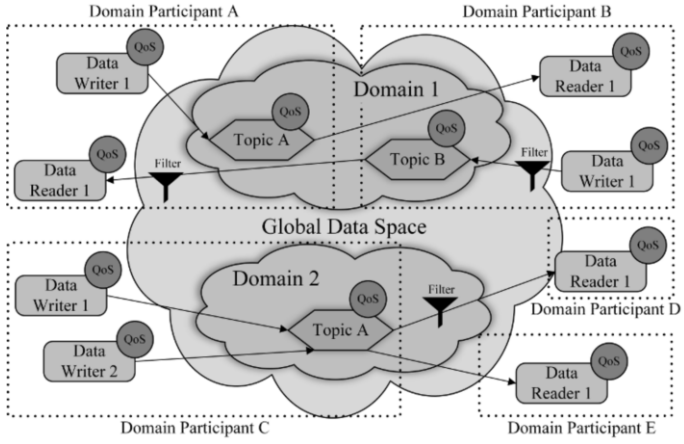

Fig. 1. The DDS architecture.

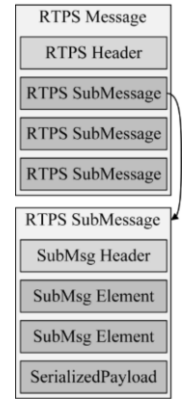

Fig. 2. The RTPS message structure.

cations locally store their required data temporarily while the rest is held in remote nodes and accessed through APIs. Based on the principles of the publish-subscribe paradigm, data is read from or written to Topics by applications using Data Reader or Data Writer entities also referred to as Endpoints. DDS Domains logically divide the data space to optimize communication and addressing. Applications can host Domain Participant entities that create and hold Data Writers, Data Readers and Topics. Furthermore, they contain security and QoS-related configurations and are identified with Globally Unique Identifiers (GUIDs). A Domain Participant is bound to a single Domain and cannot communicate with Endpoints on another Domain. A discovery module in the RTPS protocol uses preconfigured dedicated Data Readers, Data Writers and Topics to announce and collect QoS and security policies and information about new Domain Participants inside the domain. A newly created Domain Participant announces itself to a list of configured target peers over multicast. Changed properties of a Domain Participant are automatically propagated. Endpoints can also be excluded from discovery announcements if network or device resources are limited.

From the point of view of an application, it has direct access to the entire part of the global data space, while in reality relevant data is remotely accessed on demand. The global data space is therefore a collection of local application stores that are available to all participants on a peer-to-peer basis. All devices are essentially connected to a virtual bus and communicate with each other without the need for a cloud or server broker. This decentralized architecture enables systems to communicate in real-time while scaling across a large amount of participants.

Real-Time Publish-Subscribe Protocol The DDS DCPS specification sets various requirements for its underlying transport protocol, such as discovery services for new devices, lack of single points of failure like centralized name servers or information brokers and QoS properties to enable fault tolerance, reliability and timeliness. However, it does not actually specify a wire protocol or the data encoding. To ensure that different DDS implementations can interoperate, the Real-Time Publish-Subscribe Protocol was utilized. RTPS defines a standard wire protocol that can take advantage of the configurable QoS settings in DDS. It was first approved as part of the Real-Time Industrial Ethernet Suite IEC-PAS-62030 by the IEC and was tailored to comply with the requirements of publish-subscribe data-distribution systems. As a result it forms a close synergy with DDS and its underlying behavioral architecture. Similar to DCPS, RTPS is specified through a PIM which defines message structures, sets of legal message exchanges and discovery functions. Even though the original specification by OMG only provides a Platform Specific Model (PSM) for UDP/IP, most commercial implementations, like Connext DDS, CoreDX or OpenSplice, adopted a TCP/IP binding as well.

As seen in Figure 2 the RTPS message structure consist of a RTPS Header, followed by a variable amount of RTPS Submessages. Each Submessage consists of a Submessage Header, Submessage Elements and optionally a Serialized

Payload. The RTPS Header identifies the RTPS protocol and its vendor. Submessages contain application data or different types of metadata information, such as acknowledgements. The specific type is identified by the Submessage Header. In the case of an acknowledgement, the Submessage Elements could be a set of sequence numbers for missing messages and the related Data Reader and Writer IDs for identification. A Submessage of type Data contains the value of an application data-object in the Serialized Payload and the data type and other relevant information in the Submessage Elements.

Quality of Service Since DDS utilizes a broker-less infrastructure, communicating entities run the risk of loosing control over the data flow and either receive unnecessary information or do not meet the reliability requirements of the application. QoS policies are therefore a very important property of the DDS architecture. As indicated in Fig. 1, each DDS entity has a set of configurable QoS parameters that provide a large number of possibilities to regulate the data flows. If certain Topics need to be updated periodically, the *Deadline* policy establishes a contract that the publishing application has to meet. Similarly, subscribing applications can set the *TimeBasedFilter* policy to prevent receiving all published messages to a Topic and only retrieve data in fixed time intervals. To name a few more examples, the *Durability* policy allows late joining Data Readers to obtain historical data from Data Writers by configuring its durability mode; the *Liveliness* policy helps applications to identify inactive entities within the DDS data space. Redundancy can be achieved by publishing data from multiple Data Writers to the same Topic. The *Ownership* policy allows to configure which publisher can update the contents of the Topic and in what order other publishers are authorized to do so if the first one fails to deliver in time.

3.2 DDS Security

Information assurance in DDS is realized within the DDS Domains, where the read and write access between Topics and Domain Participants is controlled. Application-defined security policies that affect a DDS Domain, its Topics and participants are enforced by DDS Security.

The DDS Security Model specifies five service interfaces that allow the implementation of plugins. Each of these Service Plugin Interfaces (SPIs) implement certain security functions. Combined together they provide information assurance to DDS systems. This modular approach allows for flexible deployment and makes it possible to complement existing DDS implementations without having to adapt them. DDS provides built-in plugins, namely *DDS:Auth:PKI-DH* for authentication, *DDS:Access:Permissions* for access control and *DDS:Crypto:-AES-GCM-GMAC* for message encryption and authentication. They can be used as is or can be replaced with customized implementations. In this paper however, only the built-in standardized service plugins are evaluated. The additional Logging Service Plugin provides means to log security events for a Domain Participant. The Data Tagging Service Plugin allows the addition of security labels

or data tags to messages for additional access control and message prioritization. Since they are not mandatory conformance points with DDS Security, they will not be further discussed in this paper.

Authentication Service Plugin *DDS:Auth:PKI-DH* facilitates the authentication of Domain Participants invoking operations in DDS Domains. Depending on the configuration, a Domain Participant might need to be authenticated in order to communicate in a DDS Domain. Authentication is based on a trusted Identity CA that mutually authenticates Domain Participants, either by using Rivest-Shamir-Adleman (RSA) [MKJ+16] or Elliptic Curve Digital Signature Algorithms (ECDSA) [ANS05]. Depending on the used algorithm, the public and private key-pairs of the CA and the Domain Participant shall either be 2048-bit RSA keys or 256-bit EC keys for the NIST P-256 curve [Nat13]. To calculate a shared secret, ephemeral variants of either 2048-bit Diffie-Hellman (DHE) key or 256-bit Elliptic Curve Diffie-Hellman (ECDHE) key agreement methods are used [Res99]. X.509 v3 certificates [Int16] are used for identification. They are validated with a Certificate Revocation List (CRL) [CSF+08] or/and the Online Certificate Status Protocol (OCSP) [SMA+13]. A shared Identity CA signs the identity certificates of Domain Participants. The same or a separate shared CA signs the permissions and domain governance documents which specify the access rights of a Domain Participant and the Domain's security policies respectively (Section 3.2). Regardless of the choice, the CAs will be labeled separately as Identity CA and Permissions CA throughout this paper. Before a Domain Participant is enabled it therefore needs to be configured with:

- A X.509 v3 CA certificate that defines the Identity CA.
- A X.509 v3 identity certificate that defines the Domain Participant by binding its GUID to its public key and chains up to the Identity CA.

Used for later access control (Section 3.2):

- A X.509 v3 CA certificate that defines the Permissions CA.
- A Domain governance document, signed by the Permissions CA.
- A Domain permission document, signed by the Permissions CA.

Authentication is started by a 3-message handshake protocol. If for example a Data Reader of Domain Participant A wants to read from a Topic hosted by Domain Participant B, the Domain Participant with the lowest GUID sends a handshake request.

1. If Domain Participant B initiates the protocol, the request will contain its generated DH public key (*B.dh*), its identity 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*) used for challenge-response authentication and key derivation.

2. Domain Participant A validates the identity certificate *B.cert* against the Identity CA and sends a reply with its own generated DH public key (*A.dh*), its own CA-signed identity certificate (*A.cert*), its CA-signed Domain permissions document, the received 256-bit nonce (*B.rand*), a newly generated 256-bit nonce of its own (*A.rand*) and a generated digital signature (*A.sig*) on *A.rand* and *B.rand*.
3. Domain Participant B validates *A.cert*, verifies *A.sig* and sends its own signature (*B.sig*) on *A.rand* and *B.rand*.
4. Domain Participant A finalizes the handshake after verifying *B.sig*.

Both Domain Participants use each others DH public keys (*A.dh*, *B.dh*) to compute a shared secret and proceed to hash it with the 256-bit Secure Hash Algorithm (SHA256) [EH11]. Domain Participant B generates a master salt, a random session ID and a random Initialization Vector (IV) suffix. Based on HMAC-Based Key Derivation (HKDF) [KE10], Domain Participant B creates a master sender key using the shared secret, the master salt, *A.rand* and *B.rand* in the extraction phase. The master sender key and the salt are sent to Domain Participant A using a built-in secure Data Writer and Data Reader. The sent key material is encrypted using the shared secret. As the second step of the HKDF, both Participants expand the master sender key, master salt, session ID, *A.rand* and *B.rand* into a session key. HMAC-SHA256 serves as the extraction function and as the underlying pseudorandom function (PRF) [EH11] for the expansion. The IV is formed by the IV suffix and the session ID. The session key and the IV are used to create the actual ciphertext and MACs. A crypto header precedes every encrypted message, containing the session ID and the IV suffix. For each subsequent MAC or encrypt operation under the same session key, the IV is incremented. When reestablishing a connection, the session ID is incremented and a new session key is derived from the new session ID and the old key material.

Access Control Service Plugin *DDS:Access:Permissions* implements access control mechanisms within DDS systems. As described in 3.2, a Domain Participant needs to be configured with a Domain governance and permission document, signed by the Permissions CA. The document's purposes are specified as follows:

– The Domain governance document specifies the security policies of a domain. It configures various aspects applying to the whole domain, such as the protection mode for RTPS messages (Section 3.2), whether discovered but unauthenticated Domain Participants should be able to access unprotected Topics, whether read or write access to Topics should be restricted to Domain Participants that have the proper permissions, whether only the Serialized Payload, the Submessage or the whole RTPS message (Section 3.1) sent on specific Topics should be protected and with which kind (see protection levels in Section 3.2) etc.

– The permissions document contains the identity of the Domain Participant to which the permissions apply, matching the contents of its identity certificate. Furthermore, the valid time dates for the permissions are stated. A set of rules define the permissions of the Domain Participant. Any DDS operation like joining a Domain or publishing/subscribing to certain Topics is either allowed or denied. Referring to the example in Section 3.2, after successful authentication and granted that the governance document requires Participants to have proper permissions to access any Topic, Domain Participant B would first check the permissions document of Domain Participant A whether it is allowed to access the requested Topic or not.

Cryptographic Service Plugin *DDS:Crypto:AES-GCM-GMAC* provides various levels of message encryption and authentication. It utilizes the Advanced Encryption Standard (AES) in Galois Counter Mode (AES-GCM) [Nat07] and supports 128-bit and 256-bit key sizes with a 96-bit nonce as IV. MACs are provided using Galois MAC (AES-GMAC) [Nat07]. GCM is an operation mode for symmetric key cryptographic block ciphers. It enables authenticated encryption at high throughput with low cost and latency. Because the block ciphers are generated with counters, they can be encrypted and decrypted in arbitrary order. This is important for DDS, as Data Readers might not receive all data samples written by a Data Writer when content filtering QoS policies are applied. Furthermore, this allows parallel processing and pipelining of block encryption and decryption operations, optimizing it for the real-time communication requirements of DDS. Each AES-GCM transformation produces a ciphertext and a MAC using the same secret key. AES-GMAC is essentially an authentication-only variant of AES-GCM which only generates a MAC if no plaintext is selected for encryption.

The Domain governance document (3.2) specifies the protection mode of RTPS messages within a Domain. Depending on the security requirements of the network environment and the available computational resources, RTPS messages can be secured as a whole, on Submessage or even only on Payload level (Section 3.1). For that, different protection modes are provided:

None – indicates no cryptographic transformation.

Sign – indicates the cryptographic transformation shall only be a message authentication code (MAC) without encryption.

Encrypt – indicates the cryptographic transformation shall include encryption followed by a MAC that is computed on the ciphertext.

3.3 TLS and DTLS

The Transport Layer Security protocol (TLS) and the Datagram Transport Layer Security protocol (DTLS) are applied over TCP and UDP respectively, in order to provide confidentiality, integrity and authenticity. In this work the focus lies on the security features of TLS version 1.2 [DR08] and DTLS version 1.2 [RM12].

Transport Layer Security The Transport Layer Security protocol is used to apply security for communication between client/server applications over a reliable transport protocol and operates independently of application layer protocols. TLS utilizes two layers for establishing the secure connection, namely the TLS handshake protocol and TLS record protocol. The TLS handshake protocol ensures authentication of a peer's identity, negotiation of the protocol version, cryptographic algorithms and computation of a shared secret. It utilizes X.509 v3 certificates and trusted CAs that authenticate servers and optionally clients. The TLS record protocol uses the keys generated from the handshake, to ensure data confidentiality and integrity. The record protocol uses symmetric-key cryptography for data encryption and ensures integrity through MACs. TLS supports a variety of cipher suites listed in [DR08,SCM08]. They determine the algorithms used for key exchange, authentication and symmetric-key encryption.

If a client wants to establish a mutually authenticated and secure communication channel to a TLS-enabled server, the client will initiate the TLS handshake protocol as shown below. For the purpose of comparing the security features of DDS and TLS/DTLS, the used cipher suite shall be TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 [SCM08], so that it is configured similarly to the specified authentication and cryptographic plugins of DDS.

1. The client sends a ClientHello message to the server containing the requested TLS protocol version, cipher suite and a 256-bit nonce (*C.rand*) for challenge-response authentication and key derivation.
2. The server sends a ServerHello message which contains the chosen TLS protocol version, the cipher suite, the session ID and the server's nonce (*S.rand*). It proceeds to send its CA-signed certificate (*S.cert*). As an ephemeral DH key-exchange was negotiated, the server additionally sends its DH public key (*S.dh*) and appends a generated digital signature (*S.sig*) on *S.dh*, *S.rand* and *C.rand*. It concludes the response with a ServerHelloDone message.
3. The client validates *S.cert* and verifies *S.sig*. It then sends its own CA-signed certificate (*C.cert*) and his DH public key (*C.dh*) to the server as well as a generated signature *C.sig* on all previous handshake messages including *S.rand* and *C.rand*.
4. The server completes the handshake after validating *C.cert* and verifying *C.sig*.

In case of TLS version 1.2, the hash algorithm specified in the cipher suite is the base for the PRF involved in the key derivation. Server and client compute a premaster secret using each others DH public keys (*C.dh* and *S.dh*). A master secret is generated from the premaster secret, *C.rand* and *S.rand* using HMAC-SHA256. Further key material is computed from the master secret, *C.rand* and *S.rand* using a PRF similar to the one used in the expansion phase of the HKDF. The key material includes the session keys that are used by the record layer protocol to create MACs and the ciphertext for both directions individually. The IV consists of an implicit and explicit nonce. The implicit nonce is created as part of the key material. The explicit nonce is generated by the sender and is carried by each TLS record. For every invocation of the GCM encryption function under the same session key, the explicit nonce shall be incremented. A TLS session can be resumed by exchanging a new set of *C.rand*, *S.rand* and generating new key material.

Datagram Transport Layer Security The Datagram Transport Layer Security protocol adapts the TLS protocol to unreliable transports like UDP. Similar to TLS, it also utilizes the handshake and the record protocol with additional features to ensure the reliable delivery of handshake messages during session negotiation. These include the introduction of message and record sequence numbers to handle re-ordering and losses during the handshake or record message exchange respectively. Furthermore, message loss is handled with retransmission timers. The cipher suites used in DTLS are adapted from TLS as stated in [RM12].

4 Evaluation

In this section DDS and TLS/DTLS are compared with regard to the IT security requirements defined in Section 2.

Authenticity: Both DDS and TLS/DTLS make use of a PKI, where communicating parties are authenticated with X.509 v3 certificates, signed by a trusted

Table 1. Comparison of the built-in DDS Security plugins and common TLS/DTLS cipher suites

Requirements	TLS/DTLS	DDS
Authenticity	PKI with X.509 certificates RSA or ECDSA and DHE or ECDHE for authentication and key exchange	PKI with X.509 certificates RSA or ECDSA and DHE or ECDHE for authentication and key exchange
Confidentiality	AES-GCM	AES-GCM
Integrity	AES-GMAC	AES-GMAC
Non-repudiation	None	Possible with alternative plugins
Availability	None	QoS-policies to enable redundancy
Access Control	None	Permissions and Governance document signed by a CA

shared CA. The DDS:Auth:PKI-DH plugin offers either RSA or ECDSA for certificate authentication and relies on DHE or ECDHE for the key exchange. Similar configurations can be achieved with common TLS/DTLS cipher suites, namely:

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Further comparisons in this section and in Table 1 will assume the usage of one of these TLS/DTLS cipher suites. The use of 2048-bit keys for RSA and DHE or 256-bit EC keys for ECDSA and ECDHE conforms with the state of the art cryptographic security standards [BR18]. Furthermore, the availability of ephemeral variants of the DH key exchange provide perfect forward secrecy. The handshake of DDS closely resembles the TLS handshake protocol with minor differences regarding the key derivation and key exchange. In TLS/DTLS, both sides derive their key material from the premaster secret. In DDS, the handshake initiator derives the key material, encrypts and sends it to the other participant using the shared secret. Key derivation in DDS is based on the HKDF. In the case of not uniformly distributed keying material, an attacker could gain partial knowledge about it (e.g. exchanged DH values). HKDF uses the extract phase to eliminate the risk of having a dispersed entropy of the input keying material and concentrates it into a cryptographically strong key first to expand upon it later. TLS/DTLS utilizes a similar scheme with a differently implemented PRF and without the preceding extraction phase. However, the transformation of the premaster secret into the master secret serves a similar purpose. Nevertheless, the new TLS version 1.3 [Res18] adopted the HKDF approach to standardize the key derivation and conform with the current security standards.

Confidentiality: AES-GCM with either 128-bit or 256-bit keys is supported by DDS and TLS/DTLS and is currently considered to be a secure encryption and decryption standard [BR18]. This Authenticated Encryption with Associated Data (AEAD) cipher suite provides confidentiality, integrity and authenticity. DDS fully utilizes the AEAD functionality by using different RTPS message protection modes. Among other configuration options, it is possible to use the "authenticated encryption" on the payload and consider the header of a message as the "associated data" which is only equipped with a MAC. This gives the payload confidentiality, the header integrity and both authenticity.

Integrity: AES-GCM natively provides encrypt-then-MAC functionality. By applying a keyed-hash function on the ciphertext, its data integrity is ensured. TLS initially only made use of the MAC-then-encrypt construct which was regarded as secure at the time of its original specification. It is known today that malleable ciphertext allows attackers to alter messages without breaking the integrity of the encryption. The encrypt-then-MAC approach identifies invalid or corrupted ciphertext without having to waste resources on decrypting the message first. The introduction of AEAD cipher suites solved these concerns. In case

no data encryption is required, the nature of the AES-GCM allows to only generate MACs using the same key, referred to as AES-GMAC. While AEAD cipher suites are optionally supported by TLS 1.2, in TLS 1.3 their use is mandated and non-AEAD ciphers are not supported.

Non-repudiation: Neither TLS/DTLS nor DDS digitally sign messages outside of the initial handshake. However, the DDS specification states that alternative plugin implementations can make use of digital signatures for regular data exchange. The RTPS messages already have the necessary structure to carry additional signatures, as they could simply replace the MACs.

Availability: TLS/DTLS does not provide any means to guarantee availability. The DDS Security plugins also do not consider availability assurance, however the inherent DDS architecture and its QoS policies enable various possibilities to create redundant communication patterns and ensure the availability of data access.

Access Control: DDS makes use of permissions and governance documents that specify access rights to individual Topics and general security policies within a whole Domain. The documents are legitimized by a CA that is part of the PKI. TLS/DTLS is unable to provide such functionality, as it lacks the same tight integration into the DDS architecture.

5 Conclusion

Before the DDS Security specification was published, the usage of TLS/DTLS was best practice. DDS communication would have simply been layered on top of it. Since TLS/DTLS operates on the transport layer in the ISO/OSI reference model, all application data is encrypted according to the configured cipher suite. Whenever two Endpoints attempted to communicate, a separate TLS/DTLS session was created between the two DDS-enabled applications. However, the DDS Security specification introduced a new plugin-based security approach that was tightly knit into the existing DDS core architecture. While the basic mechanisms like authentication, key exchange and encryption were addressed with similar standards and approaches as in TLS/DTLS, more sophisticated features like access control could only be realized by integrating it into the DDS Core specification. Furthermore, availability, even though not explicitly part of the DDS Security specifications, can be ensured using the decentralized nature of the DDS communication topology. Together with the big variety of QoS parameters, redundant communication patterns can be established to avoid single point of failure scenarios. Another big difference to TLS/DTLS is the ability of DDS to separately configure the protection modes for the underlying communicating entities and their logical address space. Each Topic can be protected differently, so that data being published and subscribed is either encrypted, only signed or not secured at all. The DDS-specific RTPS messages itself can be protected on different levels and in different configurations. E.g. encrypting the payload and submessages and providing MACs for the message header or encrypting the message as a whole. The Domain address space can be differently configured by

specifying which entities need authentication and how much access is granted to them individually. While in TLS/DTLS each individual connection between two applications is secured separately with different master secrets that first need to be derived, DDS reuses key material that is once established for multiple connections. Two Domain Participants can host multiple Data Readers and Data Writers and reuse the same master sender key to derive session keys for the different connections. When adding up the various configuration possibilities of DDS Security, its advantages over TLS/DTLS become clear. DDS promises to be a highly scalable middleware that provides low-latency communication for distributed systems. A security architecture that can be tailored by the user to its own use-case is therefore essential to achieve these advertised goals, while still upholding current security standards. In environments where applications scale across powerful server nodes and resource-constrained embedded devices with networks where bandwidth capacity can be scarce, the security mechanisms and the messages on the wire need to be customizeable in complexity and size. TLS/DTLS has a large set of cipher suites but is unable to provide the same level of adaptability as the service plugins in DDS.

In this work the basic security requirements for industrial applications were defined and the the security functions of DDS Security and TLS/DTLS were discussed and evaluated. The results have shown that both make use of the same state of the art mechanisms to provide confidentiality, integrity and authenticity but TLS/DTLS lacks the ability to provide the needed customization that the highly scalable DDS architecture requires to fulfill its advertised objectives. Namely low-latency, low-overhead and simultaneous data exchange between a very large number of participants. DDS Security defines service plugin interfaces that integrate security mechanisms in a configurable manner. Their incorporation into the DDS Core architecture makes the implementation of sophisticated access control features possible that could not have been realized with a transport layer solution such as TLS/DTLS. As the actual performance differences were not evaluated in this paper, future work could provide more insight by comparing the latencies and computational efforts of running DDS over TLS/DTLS and over DDS Security in various protection modes.

References

[ANS05] ANSI X9.62. Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). 2005.

[BR18] Elaine Barker and Allen Roginsky. Transitioning the Use of Cryptographic Algorithms and Key Lengths. *Draft NIST Special Publication 800-131A*, July, 2018.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Technical report, IETF - RFC5280, may 2008.

[DR08] T Dierks and E Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. Technical report, IETF - RFC 5246, aug 2008.

[EH11] D. Eastlake 3rd and T. Hansen. US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). Technical report, IETF - RFC6234, 2011.

[FSK+18] Maxim Friesen, Kai Steinke, Gajasri Karthikeyan, Lukasz Wisniewski, Stefan Heiss, and Karl-Heinz Niemann. Sichere Middleware-Lösungen für die Industrie 4.0: Eine IT-Sicherheitsanalyse aktueller Kommunikationsansätze. In *AUTOMATION 2018 - Der Leitkogress des Mess- und Automatisierungstechnik*, 2018.

[Int16] International Telecommunication Union. Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. *SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY*, November, 2016.

[KE10] H. Krawczyk and P. Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). Technical report, IETF - RFC5869, 2010.

[MKJ+16] K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch, and RSA. PKCS #1: RSA Cryptography Specifications Version 2.2 Abstract. Technical report, IETF - RFC3447, nov 2016.

[MLD09] Wolfgang Mahnke, Stefan Helmut Leitner, and Matthias Damm. *OPC Unified Architecture*. 2009.

[Nat07] National Institute of Standards and Technology. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. *NIST Special Publication 800-38D*, November, 2007.

[Nat13] National Institute of Standards and Technology. Digital Signature Standard (DSS). *FIPS PUB 186-4*, (July), 2013.

[OAS15] OASIS. MQTT Version 3.1.1, dec 2015.

[Obj] Object Management Group. What is DDS?

[Obj14] Object Management Group. The Real-time Publish-Subscribe (RTPS) DDS Interoperability Wire Protocol Specification v2.2. (September), 2014.

[Obj15] Object Management Group. Data Distribution Service (DDS) v1.4. (April), 2015.

[Obj18] Object Management Group. DDS Security v1.1. (July), 2018.

[Res99] E. Rescorla. Diffie-Hellman Key Agreement Method Status. Technical report, IETF - RFC2631, jun 1999.

[Res18] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Technical report, IETF - RFC 8446, aug 2018.

[RM12] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. Technical report, IETF - RFC6347, jan 2012.

[SCM08] J. Salowey, A. Choudhury, and D. McGrew. AES Galois Counter Mode (GCM) Cipher Suites for TLS Status. Technical report, IETF - RFC5288, aug 2008.

[SMA+13] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. Rfc6960, IETF - RFC6960, jun 2013.

This paper was funded as part of the project IT_SIVA (ref.: 19117 N) of the Research Association for Electrical Engineering at ZVEI e.V. - FE, Lyoner Strasse 9, 60528 Frankfurt am Main, Germany, by the German Federal Ministry for Economic Affairs and Energy via the AiF as part of the program for the promotion of industrial joint research on the basis of a resolution of the German Bundestag.