Your systems.
Working as one.

Security Hands On

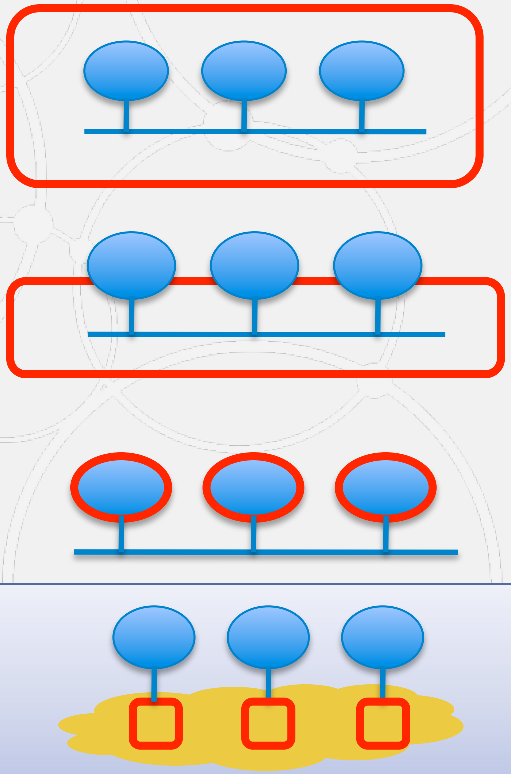
Gerardo Pardo-Castellote, Ph.D. CTO

Fernando Crespo Sanchez, Product Architect

Intro to DDS Security

Security Boundaries

- System Boundary
- Network Transport
- Host
- Data & Information Flows

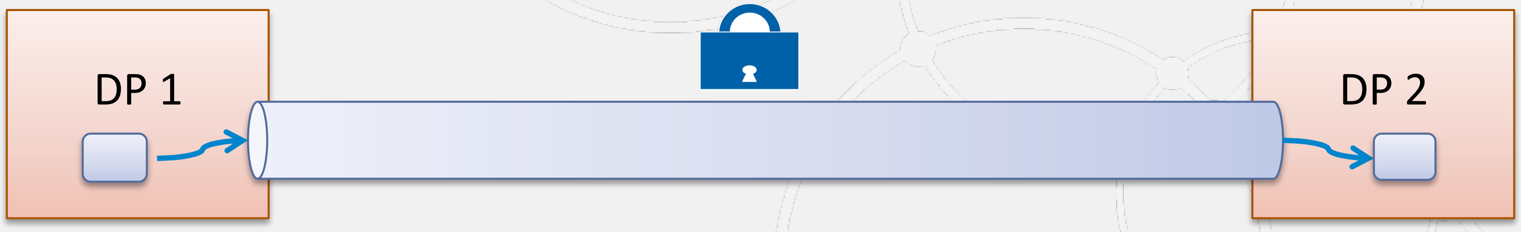


Approaches to Protect DDS

- Transport Layer Security
- Fine-Grained Security

Transport-Level Secure Data Transfer

1. Authenticate
   - Verify your identity
2. Securely exchange cryptographic keys
3. Use keys to:
   - Encrypt data
   - Add a message authentication code



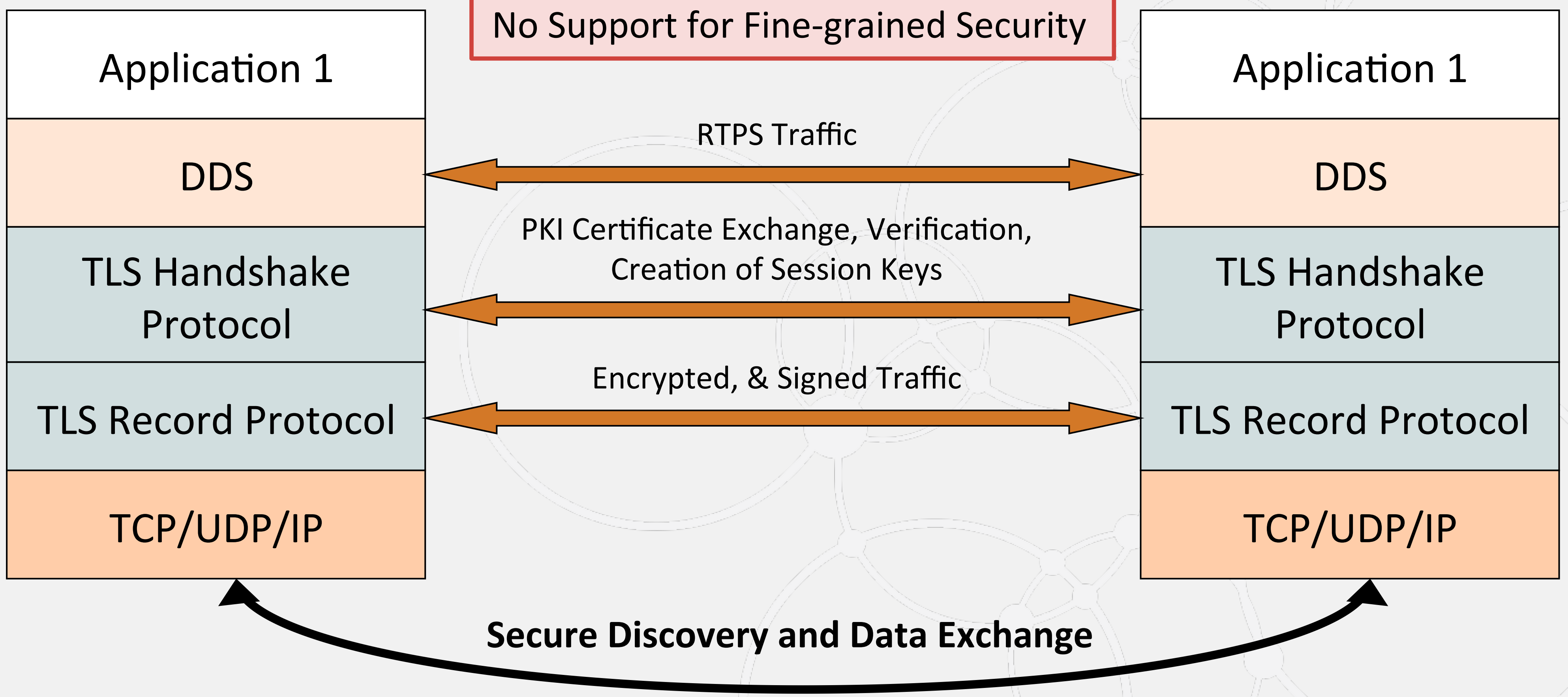
Transport-Level Secure Data Transfer In RTI Connext DDS

Three Connext DDS transports available in Connext DDS

- RTI Secure WAN Transport
  - WAN UDP transport that uses UDP hole punching to traverse NATs
  - Optional transport authentication and encryption using DTLS
- RTI Secure DTLS Transport
  - LAN UDP transport
  - Transport authentication and encryption using DTLS
- RTI Secure TCP Transport
  - WAN/LAN TCP transport
  - Optional transport authentication and encryption using TLS

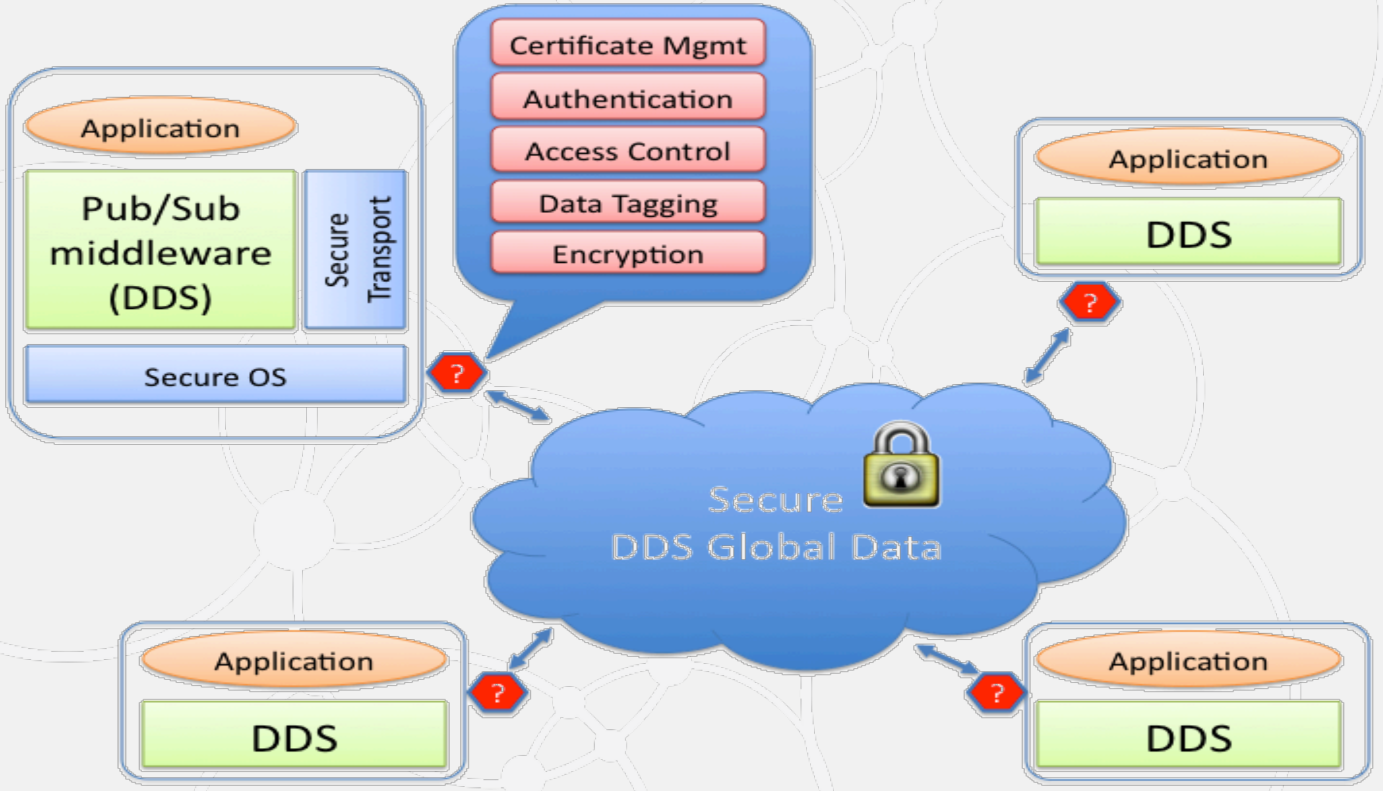
Transport Level Security

No Multicast Support
No Support for Fine-grained Security



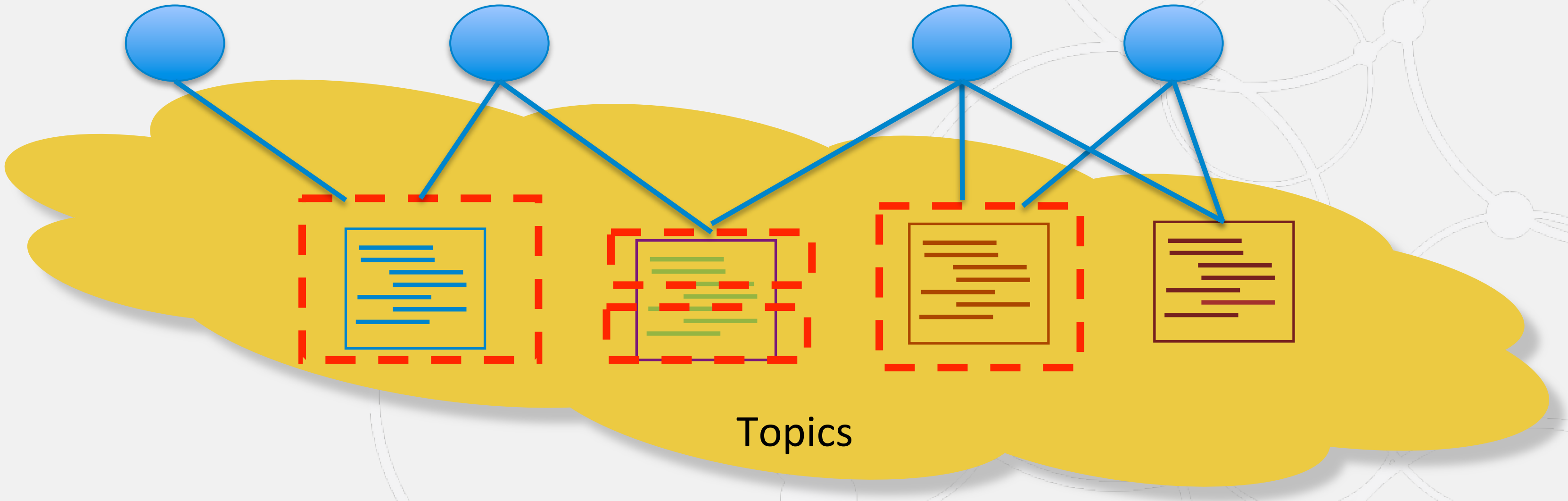
DDS Security Standard

- DDS entities are authenticated
- DDS enforces access control for domains/Topics/…
- DDS maintains data integrity and confidentiality
- DDS enforces non-repudiation
- DDS provides availability through reliable access to data



…while maintaining DDS interoperability & high performance

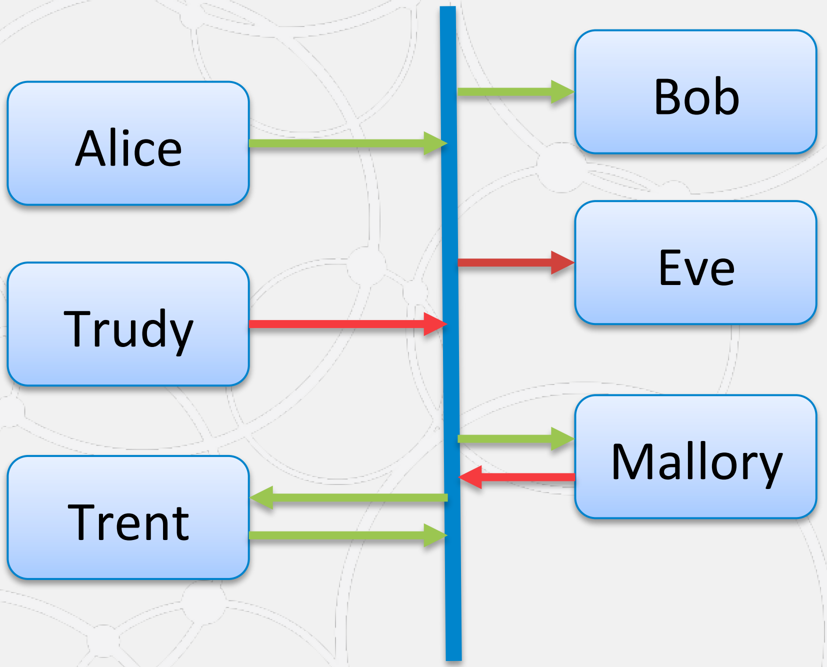
Fine-Grained Data-Centric Security



- Access control per Topic
- Read versus-write permissions
- Instance-specific permissions

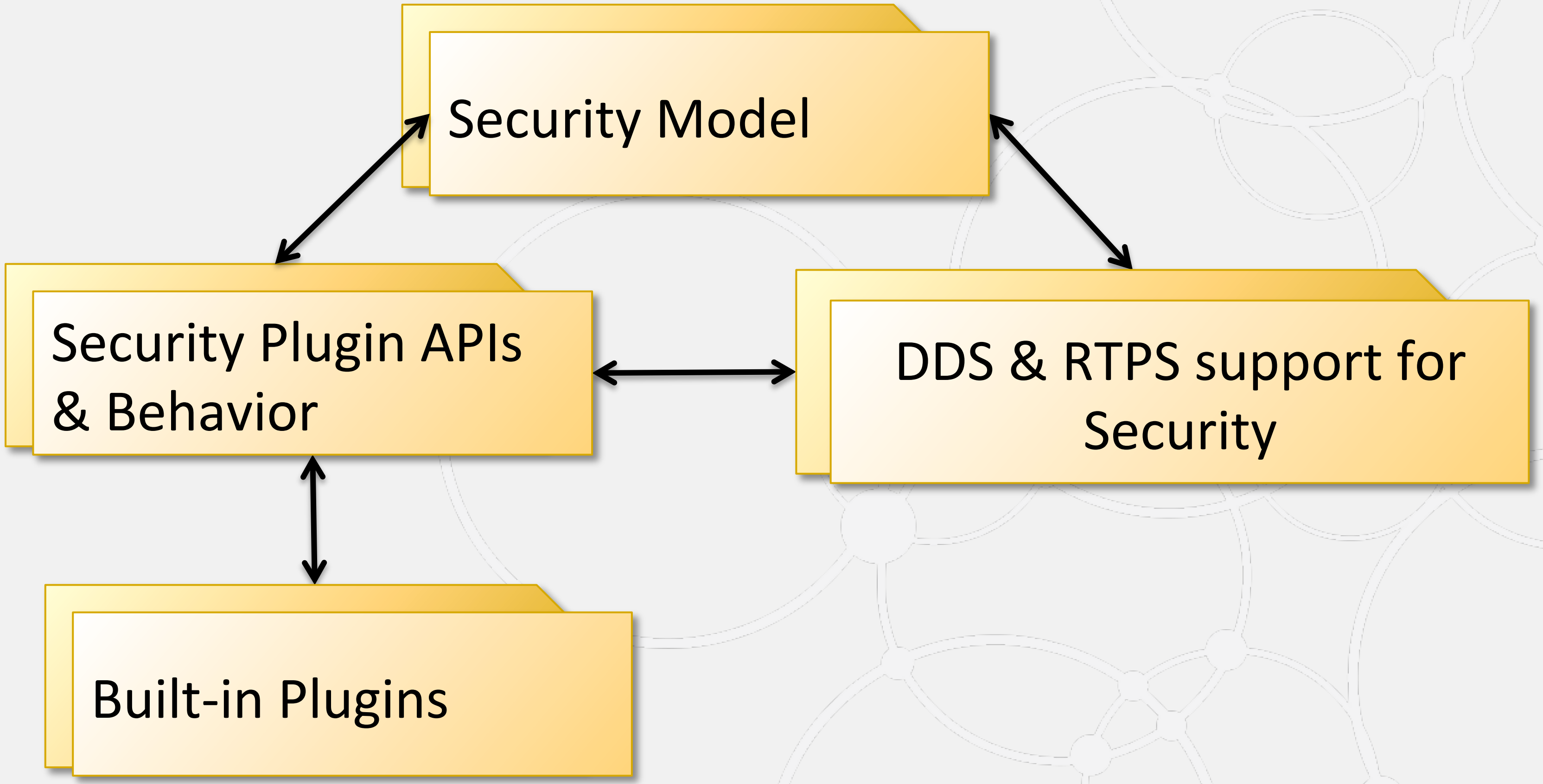
Threats

- Unauthorized Subscription
- Unauthorized Publication
- Tampering & Replay
- Insider Attack



Local machine is assumed to be trusted

DDS Security Standard Covers Four Related Concerns



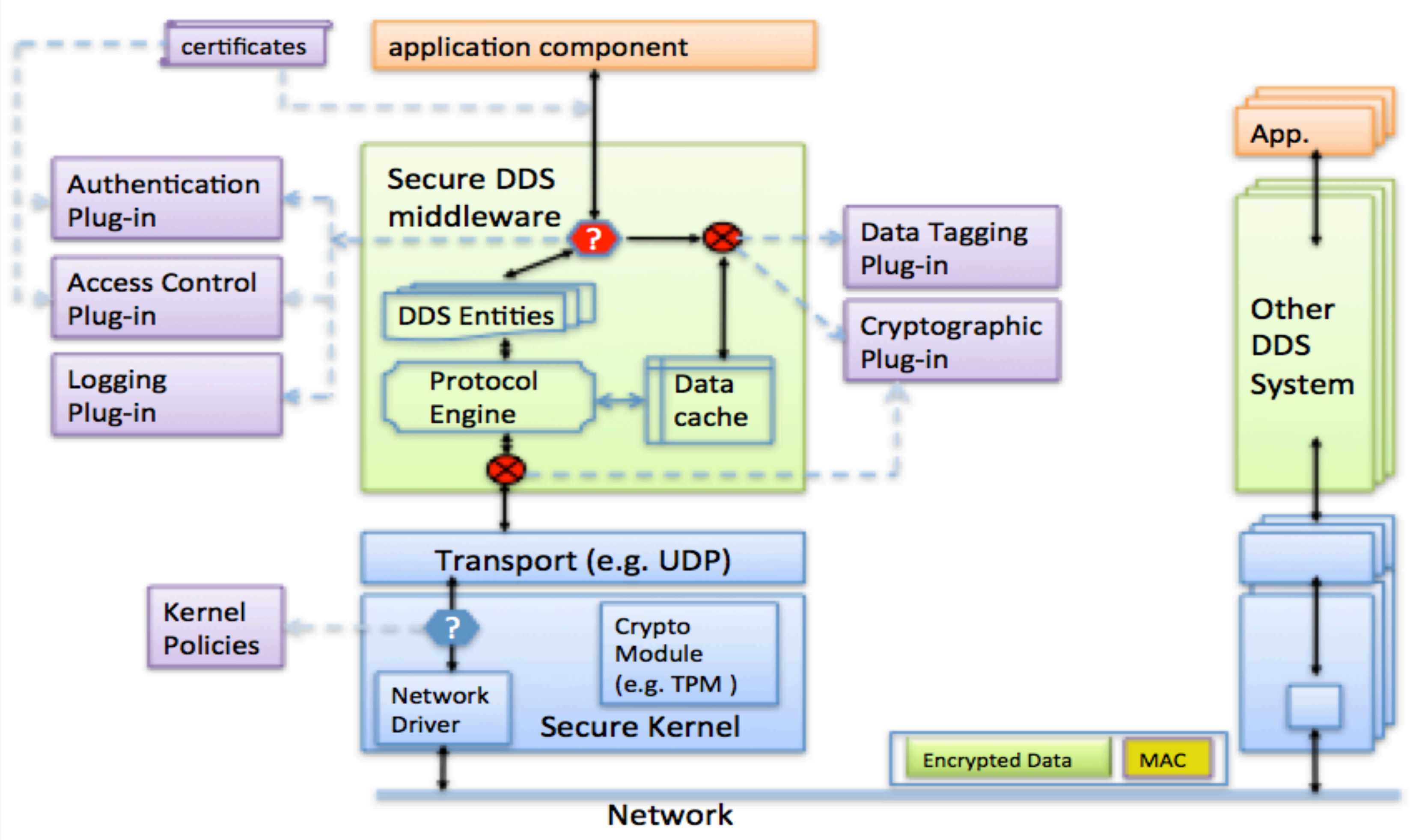
Security Model

Security Plugin APIs & Behavior

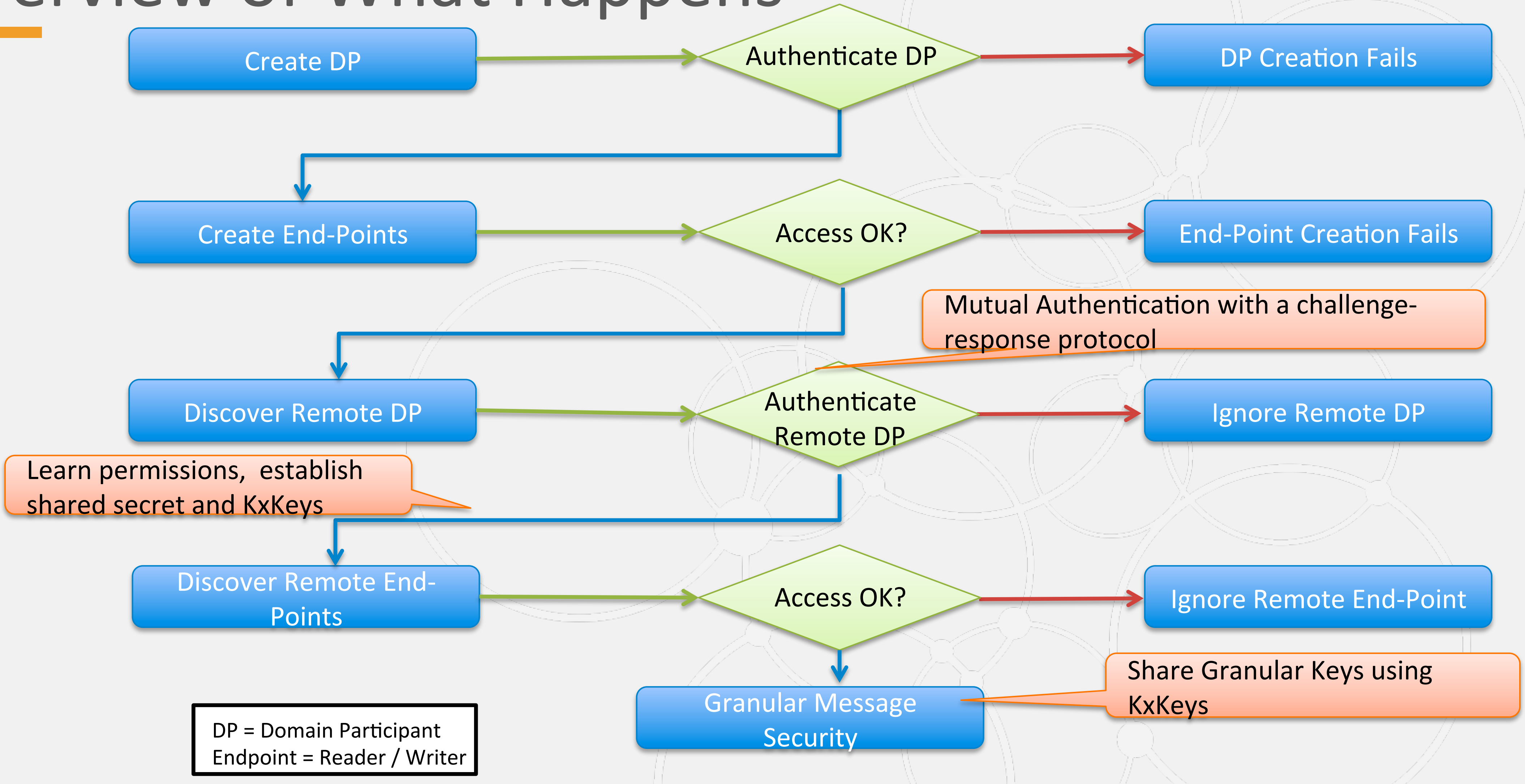
DDS & RTPS support for Security

Built-in Plugins

Pluggable Security Architecture



Overview of What Happens


Create DP
Authenticate DP
DP Creation Fails
Create End-Points
Access OK?
End-Point Creation Fails
Mutual Authentication with a challenge-response protocol
Discover Remote DP
Authenticate Remote DP
Ignore Remote DP
Learn permissions, establish shared secret and KxKeys
Discover Remote End-Points
Access OK?
Ignore Remote End-Point
Share Granular Keys using KxKeys
Granular Message Security
DP = Domain Participant
Endpoint = Reader / Writer

Pluggable Architecture

Service Plugin	Purpose	Interactions
Authentication	Authenticate the principal that is joining a DDS Domain. Handshake and establish shared secret between participants	The principal may be an application/process or the user associated with that application or process. Participants may messages to do mutual authentication and establish shared secret
Access Control	Decide whether a principal is allowed to perform a protected operation.	Protected operations include joining a specific DDS domain, reading a Topic, writing a Topic, etc.
Cryptography	Perform the encryption and decryption operations. Create & Exchange Keys. Compute digests, compute and verify Message Authentication Codes. Sign and verify signatures of messages.	Invoked by DDS middleware to encrypt data compute and verify MAC, compute & verify Digital Signatures
Logging	Log all security relevant events	Invoked by middleware to log
Data Tagging	Add a data tag for each data sample	

Built-in Plugins

SPI	Built-in Plugin	Notes
Authentication	DDS:Auth:PKI-DH	Uses PKI with a pre-configured shared Certificate Authority. DSA and Diffie-Hellman for authentication and key exchange Establishes shared secret
AccessControl	DDS:Access:Permissions	Governance Document and Permissions Document Each signed by shared Certificate Authority **Security configuration per Domain and Topic** **Access control per Domain and Topic**
Cryptography	DDS:Crypto:AES-GCM-GMAC	**Automatic key distribution** AES-128/192/256-GCM for encryption SHA1 and SHA256 for digest AES-128/192/256-GMAC for MAC **Separate keys per DW and DR** **Transparent secure multicast**
Logging	DDS:Logging:DDS_LogTopic	

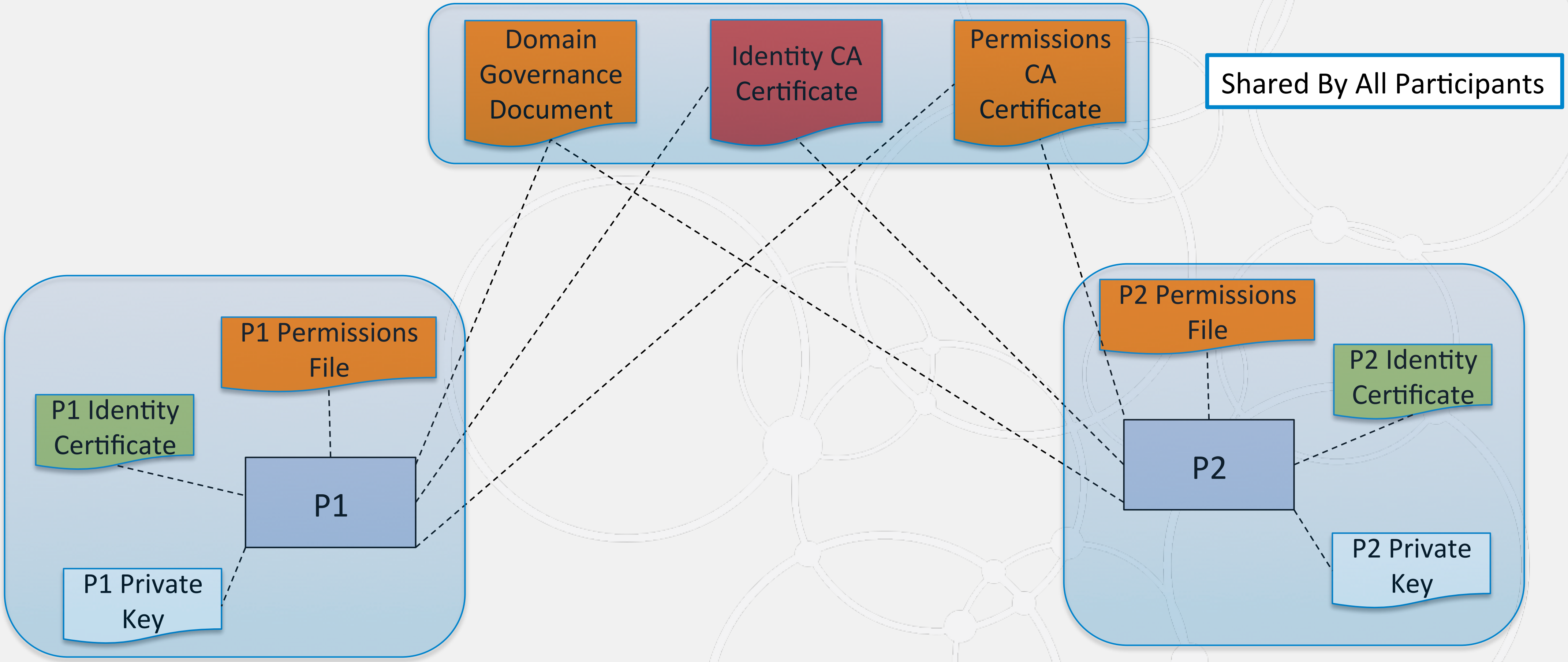
Writer Message Security

- Encryption keys & MAC keys are generated per data writer
- These keys are securely distributed to data readers
- Distribution of these keys is done using other symmetric keys derived from the shared secret
  - Key distribution is transport independent
- Different parts of messages can optionally be protected per governance policy
- Data Delivery is independent of key distribution
  - May use any transport, including multicast

Access Control & Policy

- DDS Security allows for configuring & enforcing the privileges of each participant
  - Which domains it can join & what Topics it can read/write
- It also allows specifying & enforcing policies for the whole domain, e.g.
  - Which topics are discovered using Secure Discovery
  - Which Topics have controlled access
  - Encrypt or Sign for Secure Discovery
  - Encrypt or Sign for each secure Topic
  - What to do with unauthenticated access requests

Configuring & Deploying DDS Security



Gorvernance Document

- Specifies how a domain should be secured

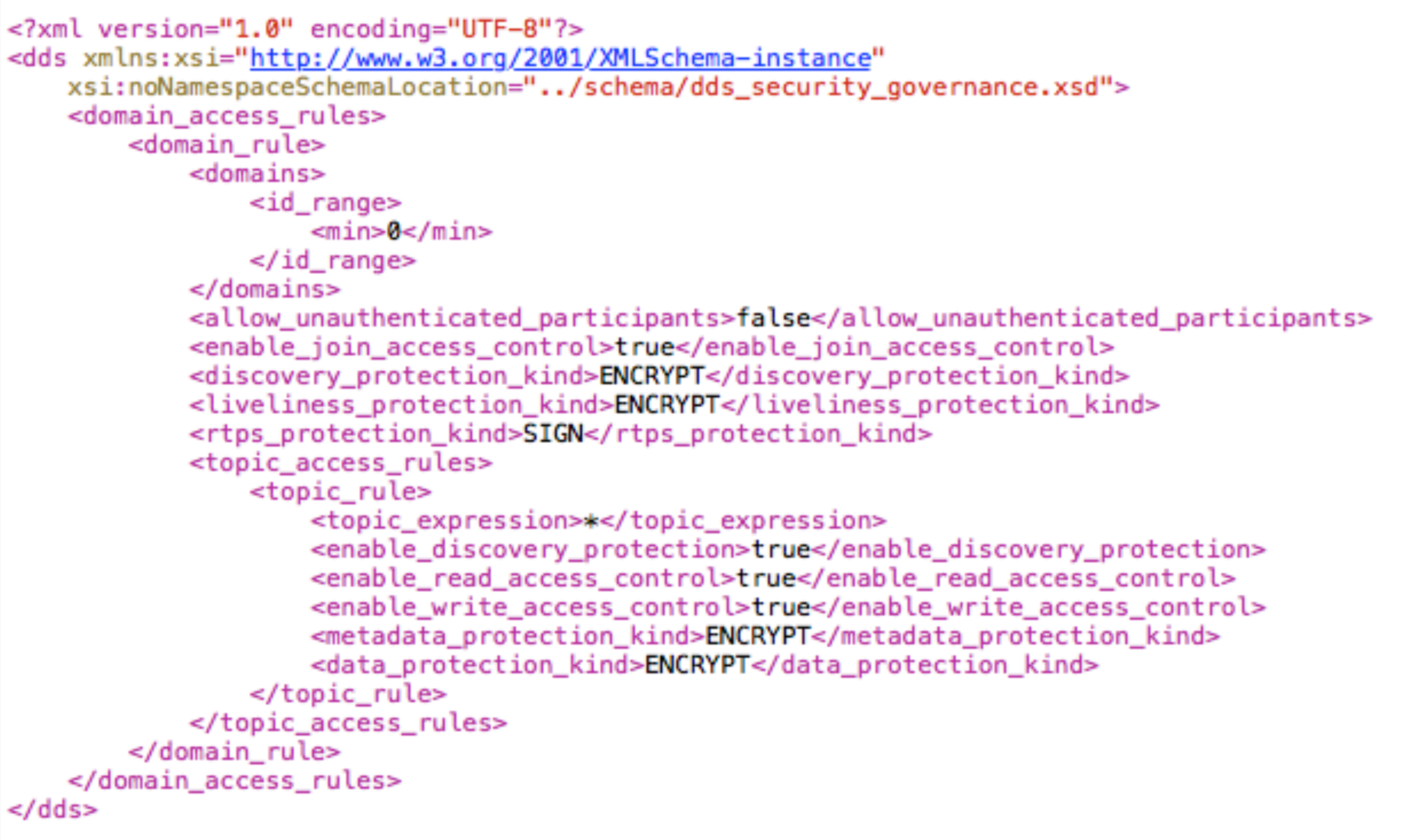
Built-in Plugins: XML Governance Document

- Specifies how a domain should be secured

- Signed by the Permissions CA

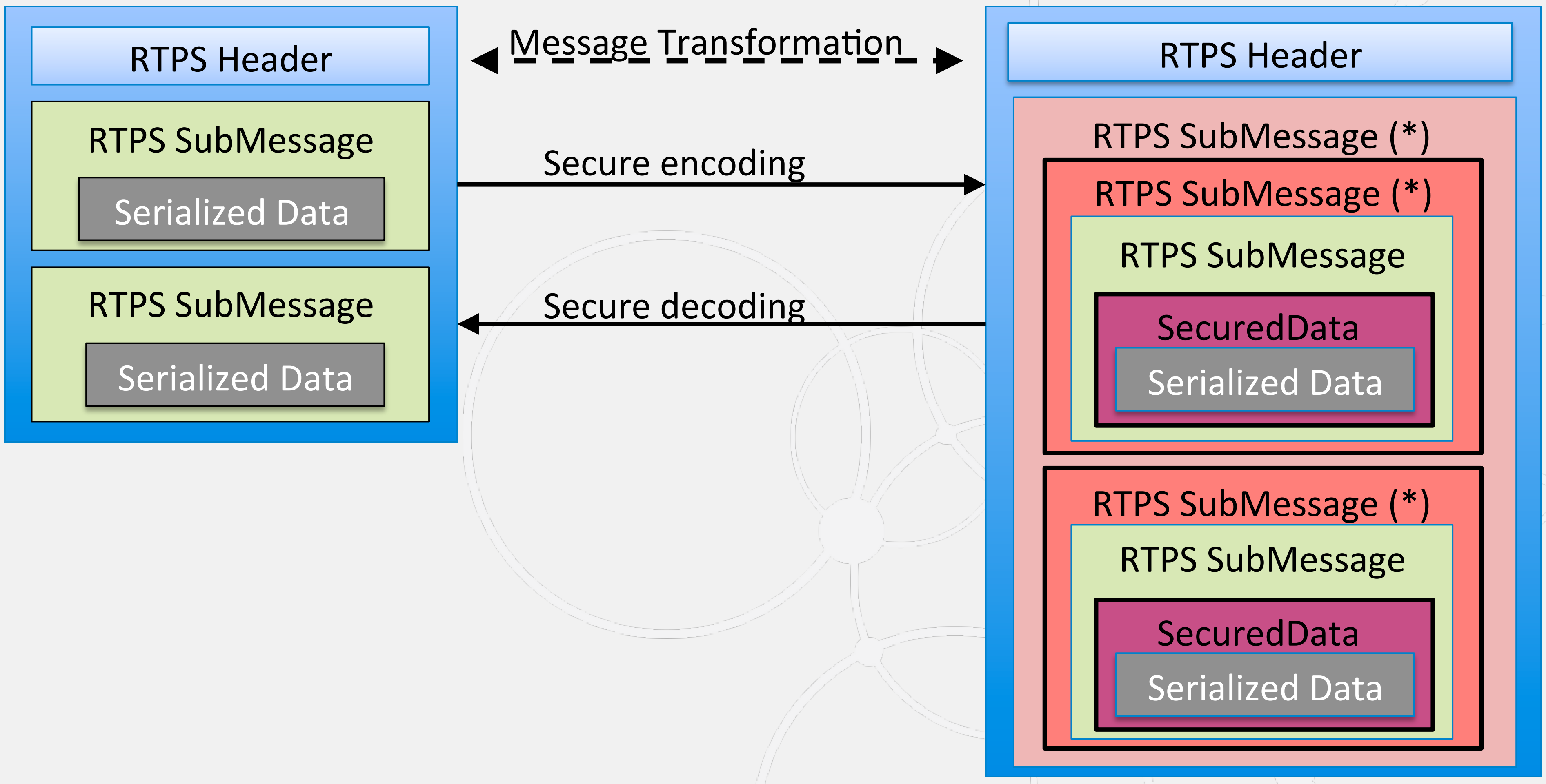
- Provided to the plugins using the PropertyQosPolicy

on the DomainParticipantQos



```
<?xml version="1.0" encoding="UTF-8"?>
<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../schema/dds_security_governance.xsd">
    <domain_access_rules>
        <domain_rule>
            <domains>
                <id_range>
                    <min>0</min>
                </id_range>
            </domains>
            <allow_unauthenticated_participants>false</allow_unauthenticated_participants>
            <enable_join_access_control>true</enable_join_access_control>
            <discovery_protection_kind>ENCRYPT</discovery_protection_kind>
            <liveliness_protection_kind>ENCRYPT</liveliness_protection_kind>
            <rtps_protection_kind>SIGN</rtps_protection_kind>
            <topic_access_rules>
                <topic_rule>
                    <topic_expression>*</topic_expression>
                    <enable_discovery_protection>true</enable_discovery_protection>
                    <enable_read_access_control>true</enable_read_access_control>
                    <enable_write_access_control>true</enable_write_access_control>
                    <metadata_protection_kind>ENCRYPT</metadata_protection_kind>
                    <data_protection_kind>ENCRYPT</data_protection_kind>
                </topic_rule>
            </topic_access_rules>
        </domain_rule>
    </domain_access_rules>
</dds>
```

Cryptographic SPI at the wire-protocol level



RTPS Header

RTPS SubMessage

Serialized Data

RTPS SubMessage

Serialized Data

Message Transformation

Secure encoding

Secure decoding

RTPS Header

RTPS SubMessage (*)

RTPS SubMessage (*)

RTPS SubMessage

SecuredData

Serialized Data

RTPS SubMessage (*)

RTPS SubMessage

SecuredData

Serialized Data

Permissions Document

- For each participant specifies:
  - What domains it can join
  - What Topics it can read/write
  - What Tags are associated with Readers & Writers

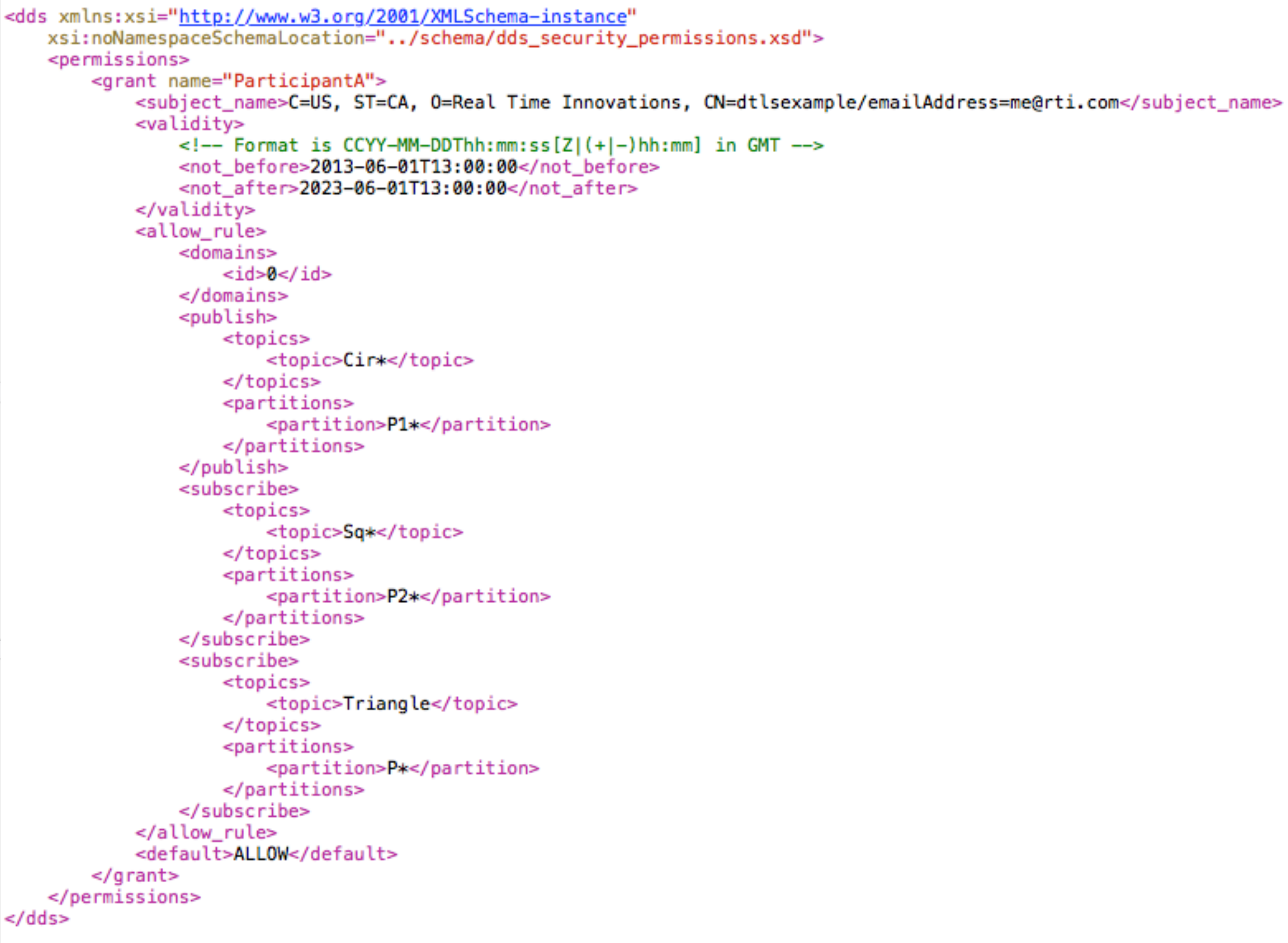
Built-in Plugins: XML Permissions Document

- Contains the permissions of the Domain Participants

- Signed by the Permissions CA

- Provided to the plugins using the PropertyQosPolicy

on the DomainParticipantQos



```xml
<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../schema/dds_security_permissions.xsd">
    <permissions>
        <grant name="ParticipantA">
            <subject_name>C=US, ST=CA, O=Real Time Innovations, CN=dtlsexample/emailAddress=me@rti.com</subject_name>
            <validity>
                <!-- Format is CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm] in GMT -->
                <not_before>2013-06-01T13:00:00</not_before>
                <not_after>2023-06-01T13:00:00</not_after>
            </validity>
            <allow_rule>
                <domains>
                    <id>0</id>
                </domains>
                <publish>
                    <topics>
                        <topic>Cir*</topic>
                    </topics>
                    <partitions>
                        <partition>P1*</partition>
                    </partitions>
                </publish>
                <subscribe>
                    <topics>
                        <topic>Sq*</topic>
                    </topics>
                    <partitions>
                        <partition>P2*</partition>
                    </partitions>
                </subscribe>
                <subscribe>
                    <topics>
                        <topic>Triangle</topic>
                    </topics>
                    <partitions>
                        <partition>P*</partition>
                    </partitions>
                </subscribe>
            </allow_rule>
            <default>ALLOW</default>
        </grant>
    </permissions>
</dds>
```

Configuration Possibilities

- Are “legacy” or un-identified applications allowed in the Domain?
  - Yes (if configured) unauthenticated applications will:
    - See the “unsecured” discovery Topics
    - Be allowed to read/write the “unsecured” Topics
- Is a particular Topic discovered over protected discovery?
  - If so it can only be seen by “authenticated applications”

Configuration Possibilities

- Is the access to a particular Topic protected?
  - If so only authenticated applications with the correct permissions can read/write
- Is data on a particular Topic protected? How?
  - If so data will be sent signed or encrypted+signed
- Are all protocol messages signed? Encrypted?
  - If so only authenticated applications with right permissions will see anything

Key Benefits

More Powerful Than Other Secure Middleware Technologies

- Standard & Interoperable
- Scalable: Supports multicast
- Fine-grain: Control Topic-level aspect
- Flexible: Build your own plugins
- Generic: Works over any transport
- Transparent: No changes to Application Code!

Secure Services and Tools

Integration with Persistence Service, Routing Service, Queuing Service, …

Secure Persistence Service

- Needs “read/write” permissions to the persisted topics
- Uses Governance file to determine how each Topic is protected.
- Uses its own per-Writer Key material
  - Stores WriterKey material in the database (encrypted)
  - Stores data in encrypted form
  - Replays data encrypted with WriterKey material
- Requires “-password” command-line to execute

Secure Routing Service

- Has 2 DomainParticipants hence:
  - 2 Identities, 2 Governance, 2 Permission files
  - Needs “read” permissions on the “Input” participant for the routed topics
  - Needs “write” permissions on the “Output” participant for the routed topics
- Output data protected according to Governance on output domain
- If Durable Writer History then
  - Stores data encrypted
  - Stores WriterKeys (encrypted) along with durable data

Secure(*) Queuing Service

- Has 1 DomainParticipant
  - Configured with Identity, Governance, Permissions…
  - Needs read permissions to the input (queued) topics e.g. “MyQueueTopicName”
  - Needs write permissions to the output topics, e.g. “MyQueueTopicName@MySharedSubscriberName”
  - Output protected according to governance for “*@MySharedSubscriberName”
- Queue producer need write permissions to “MyQueueTopicName”
- Queue consumers need read permissions to “MyQueueTopicName@MySharedSubscriberName”
- Data stored unencrypted

Secure Recording & Replay Service

- Has 1 DomainParticipant
  - Configured with Identity, Governance, Permissions
  - Needs read permissions to recorded topics
  - Needs write permissions to replay topics
  - Can store data data different ways
    - File Encryption (after recording stops it encrypts)
    - User Data can per Topic can choose:
      - NONE, Data, Data+Metadata encryption
    - Discovery data per buitin Topic can choose:
      - NONE, Data, Data+Metadata encryption

Cloud Discovery Service

- Only used to bootstrap
- Works with DDS Security without special configurations
- Can use secure transport e.g. (D)TLS

Secure Web Integration Service

- Has DomainParticipant on DDS side
  - Configured with Identity, Governance, Permissions
- Uses HTTPS on the web-client side
  - Clients identified by a Client-API-Key
  - Only clients with valid Client-API-Key can connect
    - All clients can access the Topics and Domains that have been granted DDS permissions

Secure Database Integration Service

- Has one DomainParticipant on DDS side
  - – Configured with Identity, Governance, Permissions
  - – Needs read permissions to topics stored
  - – Needs write permissions to topics monitored
- Decrypts DDS data before storing. The database itself may provide its own encryption if so configured.

Tools

- Can participate in secure domain
- Need Identity, Governance, Permissions to join DDS domain
  - Need read permissions to user Topics to display data
- Monitoring & Administration Domain needs separate security configuration
  - Need read permissions to Monitoring Topics
  - Need write permissions to Administration Topics
- Admin Console
  - Single configuration for all domains it joins
    - Single Identity
    - Governance and Permissions can vary per Domain
- Monitor UI, Ping, Spy
  - Single configuration for all domains

Not the friendliest configuration. Usability to be enhanced

Built-in Monitoring and Administration Topics

- DDS Core Monitoring Topics
- Distributed Logging: 2 Topics
- Service Monitoring and Administration Topics (see respective user's manuals)
  - Routing Service Monitoring & Admin
  - Recording Service Admin
  - Persistence Service Admin
  - Queuing Service Monitoring & Admin

Secure Labview

- Configuration per DomainParticipants
  - – Separate identities and authorities for all domains
  - – Separate governance and permissions specific to each DomainId
- Needs read/write permissions according to the topics used in each domain

Prototyper and Connector

- Uses XML application Creation to configure DomainParticipants
- Security configuration per DomainParticipant
  - Authorities, Identity, Governance, Permissions can vary per DomainParticipant

Some examples

Attack scenarios

1. Unauth Pub using DDS
2. Attack on Data (network tamper tcpwrite)
3. Attack on Meta-Data (network tamper tcpwrite)
4. Unauth Subs with DDS app
5. Unauth Network Data Snooping (using Wireshark)
6. Unauth Network Meta Data Snooping (using Wireshark)
7. Discovery snooping

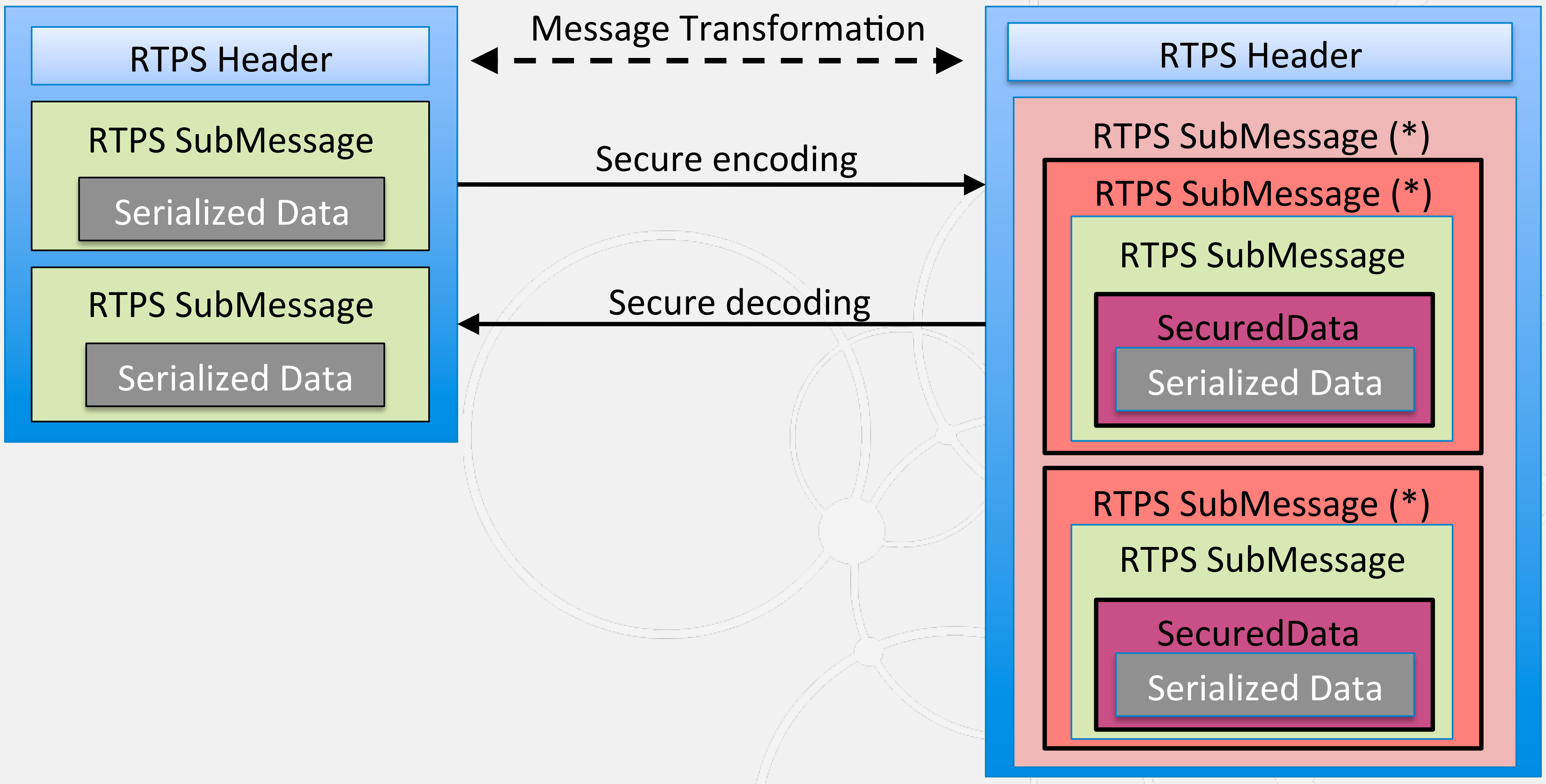
Protecting against the attacks

1. Unauth Pub using DDS
   - Require write permissions
2. Attack on Data (network tamper tcpwrite)
   - Require message signing (DATA, Submessage, RTPS)
3. Attack on Meta-Data (network tamper tcpwrite)
   - Require message signing (Submessage, RTPS)
4. Unauth Subs with DDS app
   - Require read permissions
5. Unauth Network Data Snooping (using Wireshark)
   - Encrypt (Data or Submessage)
6. Unauth Network Meta Data Snooping (using Wireshark)
   - Encrypt (Submessage)
7. Discovery snooping
   - Encrypt discovery

Performance impact

Configuration	***32B***	***1KB***	***64KB***
No Security	38 usec 580 Mbps	55 usec 975 Mbps	615 usec 990 Mbps
DDS Security. No protection	38 usec 580 Mbps	55 usec 975 Mbps	615 usec 990 Mbps
Signed RTPS (SRTPS)	45 usec 523 Mbps	65 usec 965 Mbps	690 usec 990 Mbps
SRTPS + Encrypted Data	54 usec 500 Mbps	70 usec 925 Mbps	803 usec 990 Mbps
SRTPS + Encrypted Submessage	56 usec 490 Mbps	74 usec 959 Mbps	808 usec 990 Mbps
SRTPS + Encrypted Submessage + Encrypted Data	58 Usec 480 Mbps	77 usec 917 Mbps	916 usec 990 Mbps

Cryptographic SPI at the wire-protocol level


RTPS Header
RTPS SubMessage
Serialized Data
RTPS SubMessage
Serialized Data
Message Transformation
Secure encoding
Secure decoding
RTPS Header
RTPS SubMessage (*)
RTPS SubMessage (*)
RTPS SubMessage
SecuredData
Serialized Data
RTPS SubMessage (*)
RTPS SubMessage
SecuredData
Serialized Data