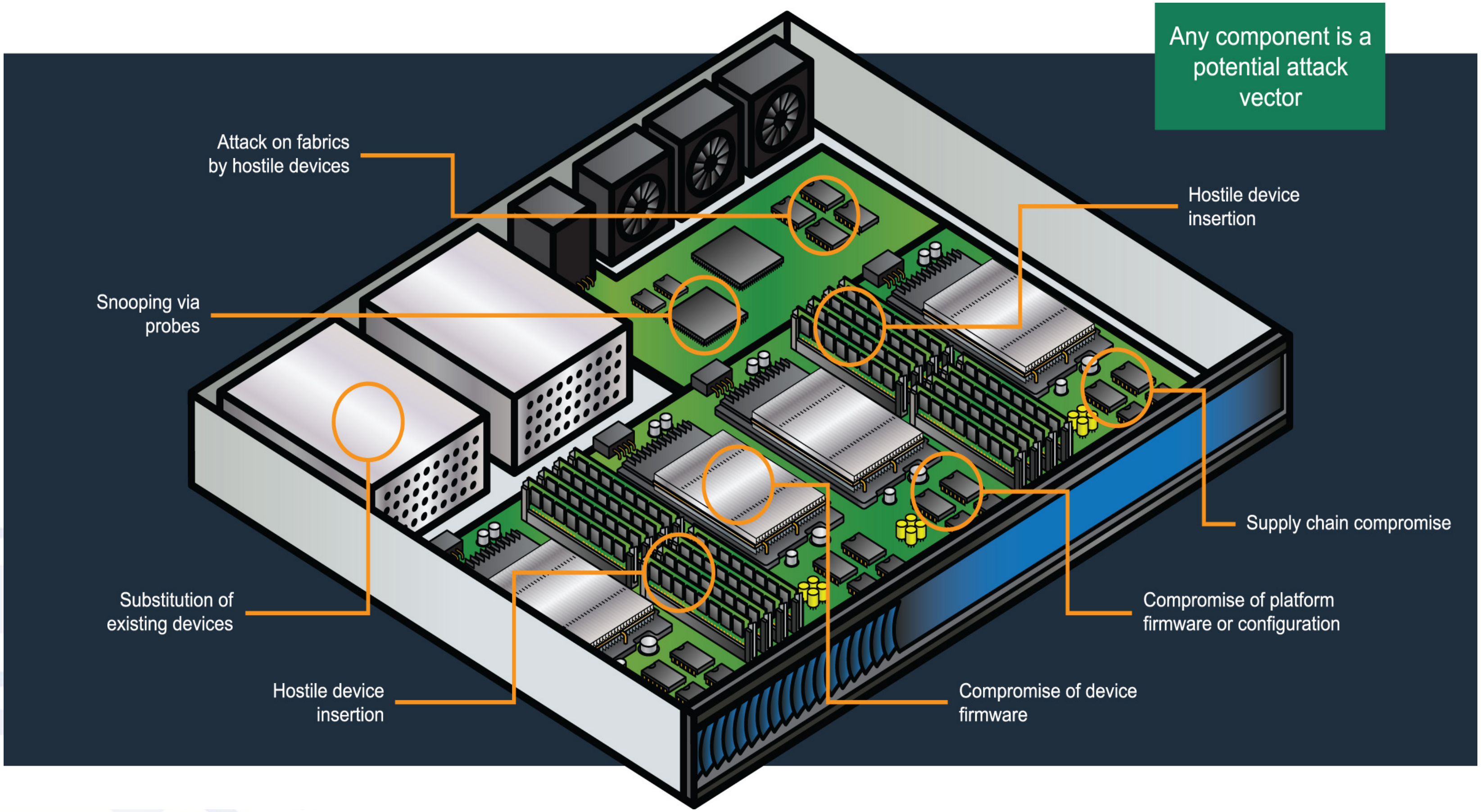
SPDM 1.3 and Beyond

August 2023

Disclaimer

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change without notice. The standard specifications remain the normative reference for all information.
- For additional information, see the DMTF website.
- This information is a summary of the information that will appear in the specifications. See the specifications for further details.

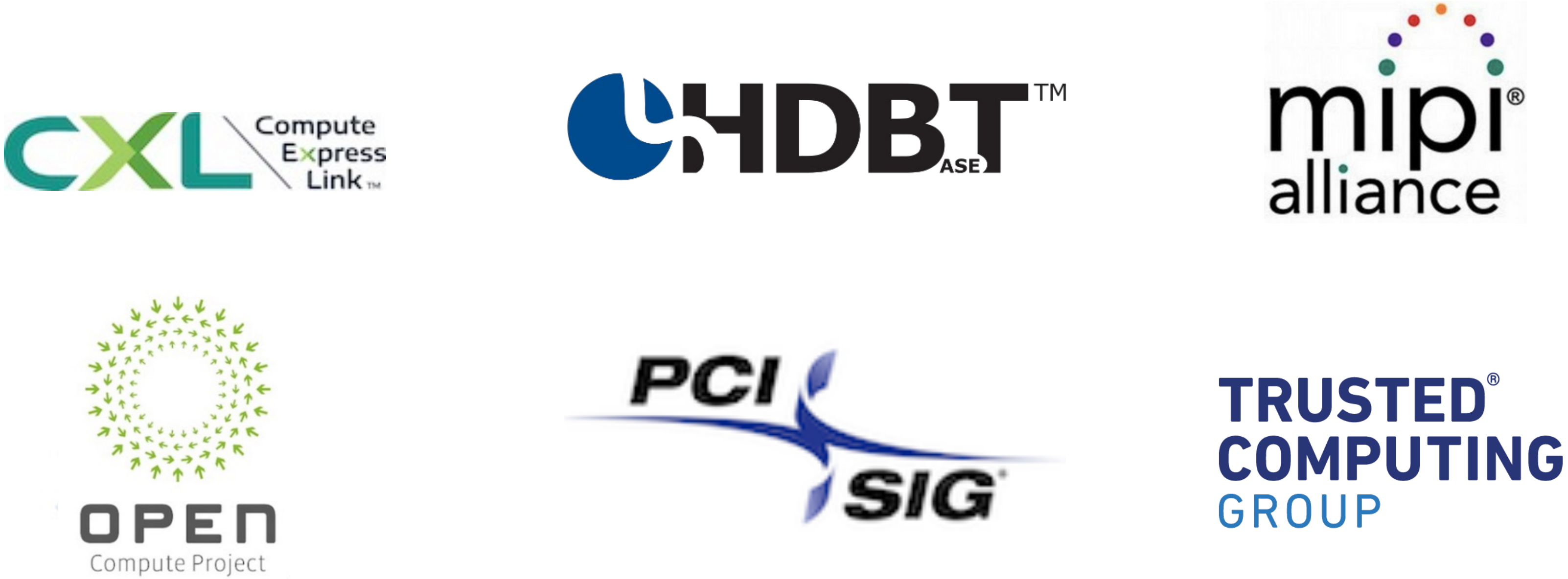
Component Threat Vectors


Any component is a potential attack vector
Attack on fabrics by hostile devices
Hostile device insertion
Snooping via probes
Supply chain compromise
Substitution of existing devices
Compromise of platform firmware or configuration
Hostile device insertion
Compromise of device firmware

SPDM's Overall Goals

- All SPDM features fall into at least one of following main goals:
- Device Attestation and Authentication
  - The ability to attest various aspects of a device such as firmware integrity and device identity
- Secure Communication over any Transport
  - Provide the ability to secure communication of any data or management traffic over any transport
  - Work with industry partners to ensure data in-flight is secure for all parts of the infrastructure (e.g. storage, network fabrics, etc.)

Alliance Partners and Adopters



SPDM Feature Summary (2023)

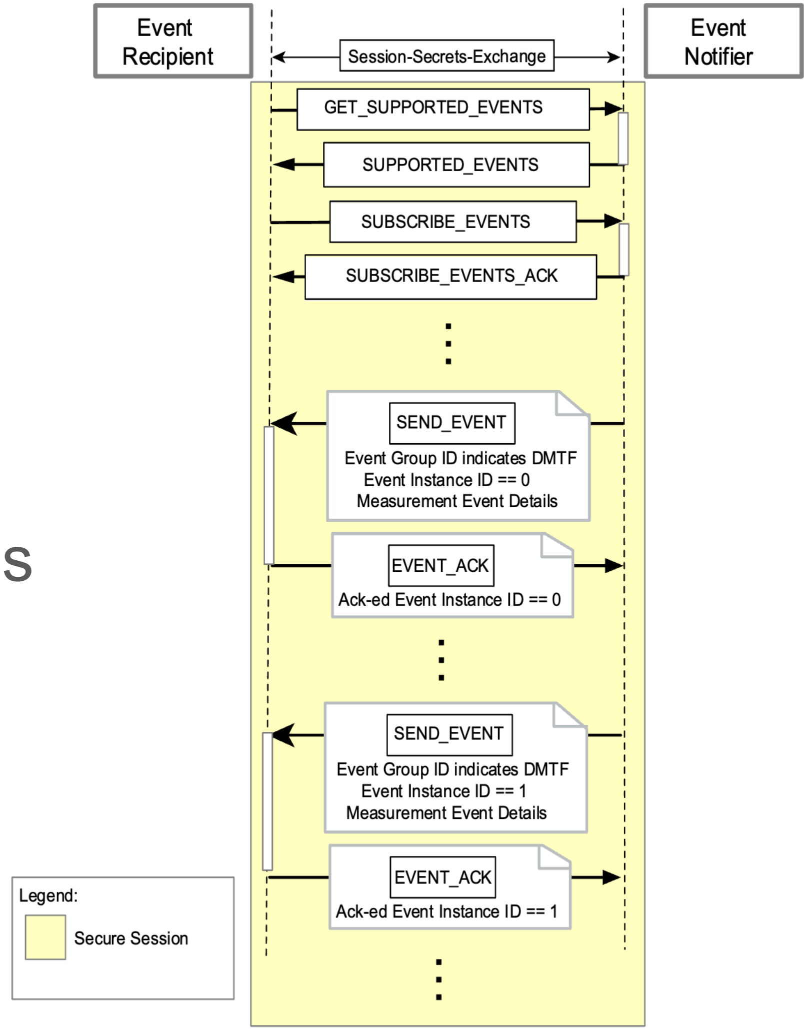
- Version 1.0:
  - Measurement Support
  - Device Attestation and Authentication
- Version 1.1:
  - Secure Session
    - Public Key Exchange
    - Symmetric Key Exchange
  - Mutual Authentication
- Version 1.2:
  - Supports installation of certificates
  - Allows for alias certificates derived from device certificates
  - Send and receive large SPDM messages (chunks)
  - Added SM2, SM3, SM4 algorithms to supported list
  - New OIDs added
  - Deprecated basic mutual authentication in CHALLENGE and CHALLENGE_AUTH

SPDM 1.3 Features

- Event Notification Mechanism
- Multi Key Support
- New Measurements
- Measurement Extension Log
- Structured Manifest format
- End Point Info

Event Mechanism

- Subscribed events
  - Interested Event Types
- All event notifications in a Secure Session
- Event Types could be extended by other standards bodies
- Can discovery supported event types and subscribe
- Notifications are ACK'd



Event Recipient

Event Notifier

Session-Secrets-Exchange

GET_SUPPORTED_EVENTS

SUPPORTED_EVENTS

SUBSCRIBE_EVENTS

SUBSCRIBE_EVENTS_ACK

SEND_EVENT
Event Group ID indicates DMTF
Event Instance ID == 0
Measurement Event Details

EVENT_ACK
Ack-ed Event Instance ID == 0

SEND_EVENT
Event Group ID indicates DMTF
Event Instance ID == 1
Measurement Event Details

EVENT_ACK
Ack-ed Event Instance ID == 1

Legend:

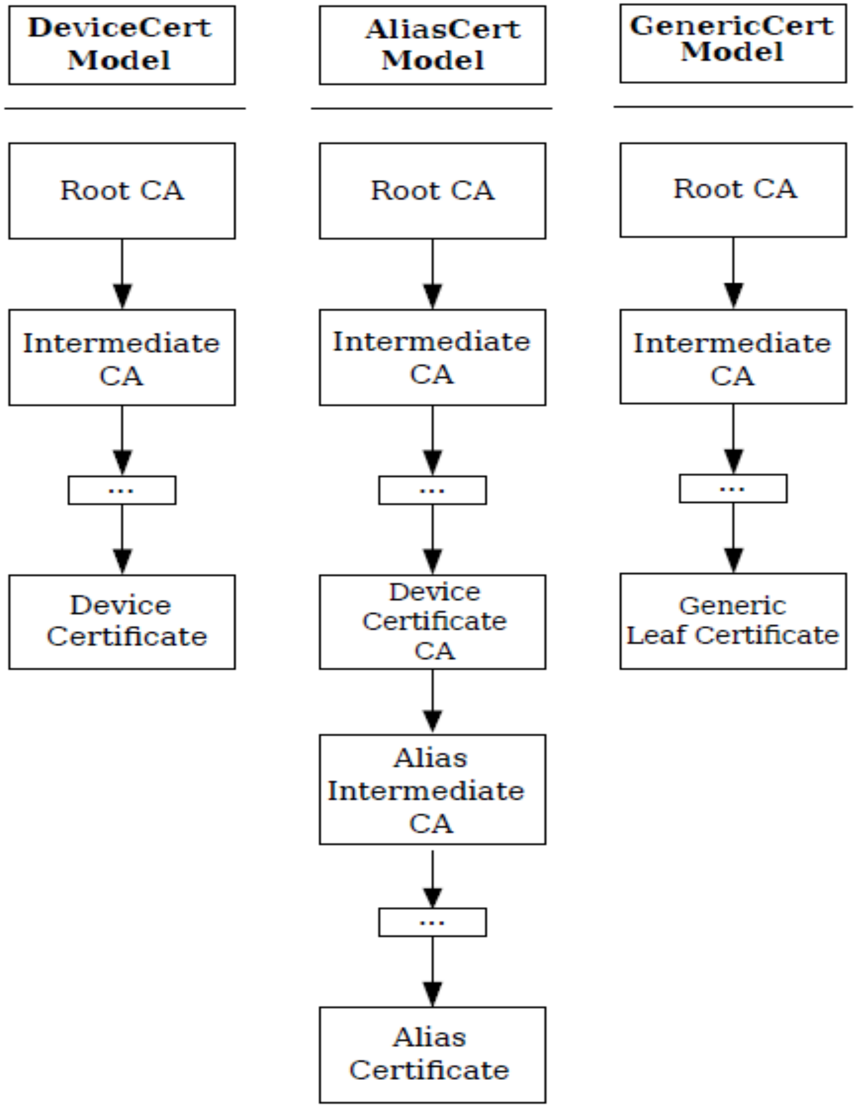
Secure Session

Multi Key Support

- Previous versions of SPDM only allowed one key pair per negotiated asymmetric algorithm
- Ability to use more than one key pair for a negotiated asymmetric algorithm
  - Up to 8 key pairs supported per asymmetric algorithm
- Every key pair could be dedicated for use case, like different key pairs for CHALLENGE and GET_MEASUREMENTS signature generations
- Requester is allowed to associate each key pair with an individual device certificate to enable one or more use cases
  - Multi Key Support enables additional use cases such as certificate provisioning in production or customer environments
  - Improved security posture
- Key pairs are identified by a unique KeyPairID

Generic Certificates Support

- What is a Generic Certificate or Certificate chain?
  - A Certificate or Certificate Chain that could not be qualified as a Device Certificate nor Alias Certificate
- New Feature
  - Generic Certificate model is introduced to support Multiple Asymmetric Keys use cases
  - Generic Certificate Model is the most flexible (or least restrictive) of the certificate models
  - Generic Certificate Model applies to certificates in slots greater than 0.
    - A Device or Alias Certificate is required in slot 0.



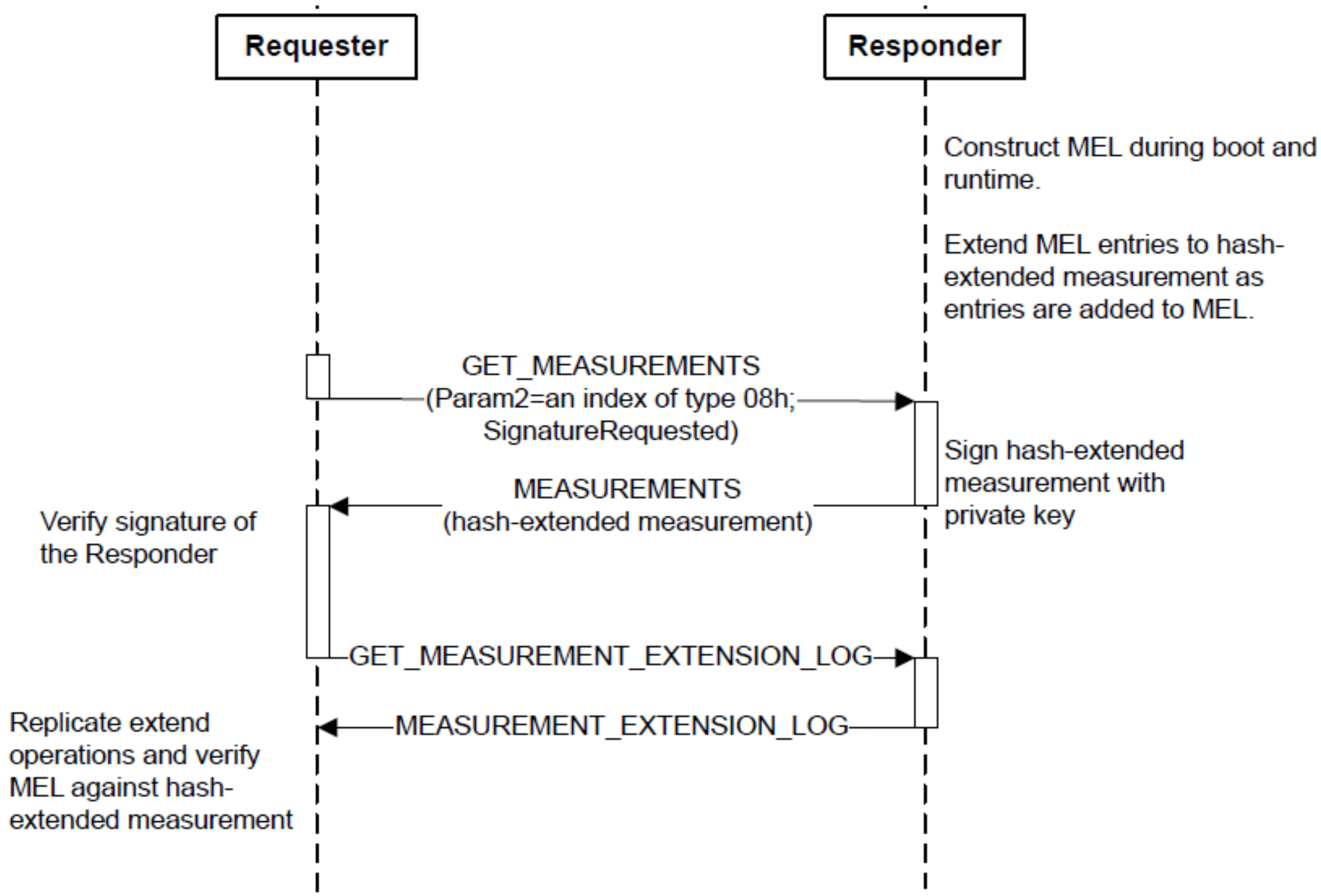
New Measurements

- `NewMeasurementRequested` field is introduced in the request attributes of the GET_MEASUREMENTS request.
  - If Responder has any changes affecting measurements that are requested by Requester but not yet applied (for example, pending changes due to a firmware update), then these new measurement values should be returned instead of current measurements (if requested using the value in the field above)
  - If there are no pending changes, then current measurements are returned regardless of the value in `NewMeasurementRequested` field
- This enables the Requester to prepare as well as to apply policy as per the system

Measurement Extension Log (MEL) and Hash-Extended Measurements (HEM)

- Responder may support reporting of measurements thru an “extend” scheme
  - Initialize HEM = HashSize bytes of 0s
  - For each extend operation, perform HEM = hash(Concatenate(HEM, DataToExtend)) for all data elements to extend
- The MEL is the collection of DataToExtend
  - Could include configuration measurements, firmware measurements, version number, etc.
  - The MEL may be preserved across resets
- An example of such a scheme is the Platform Configuration Register "extend" function in Trusted Platform Modules.
- There is a new MeasurementValueType 0x08 introduced for HEM

MEL and HEM


Requester
Responder
Construct MEL during boot and runtime.
Extend MEL entries to hash-extended measurement as entries are added to MEL.
GET_MEASUREMENTS (Param2=an index of type 08h; SignatureRequested)
Sign hash-extended measurement with private key
MEASUREMENTS (hash-extended measurement)
Verify signature of the Responder
GET_MEASUREMENT_EXTENSION_LOG
MEASUREMENT_EXTENSION_LOG
Replicate extend operations and verify MEL against hash-extended measurement

Structured Manifest format for a measurement block

- Data structure that describes the contents of other indices or contains measurements itself.
- Either Free Format or Structured
  - Free Format is implementation specific
  - Structured Format provides a Standards body or vendor-defined header, and manifest data in the format defined by the Standards body or vendor

Endpoint Info

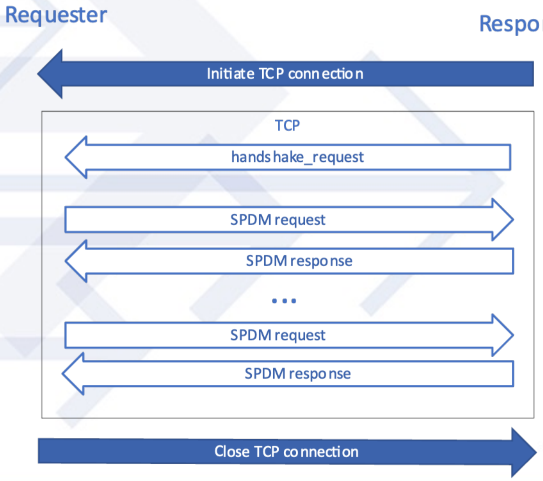
- The GET_ENDPOINT_INFO request message retrieves general information from an endpoint.
  - The SubCode parameter is used to differentiate between operations.
  - The message supports a signature.
- Currently only one Subcode is defined: ***DeviceClassIdentifier***
  - The **DeviceClassIdentifier** response returns information that can be used to identify the class of device for the Responder in question.
    - For instance, DeviceClassIdentifier could contain PCI Vendor ID and Device ID fields.

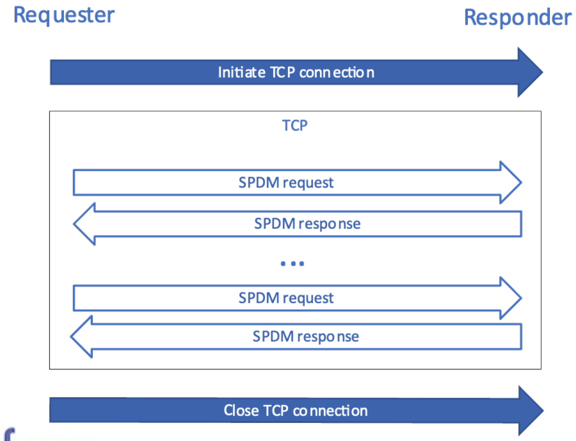
And Beyond…..

FEATURES UNDER DEVELOPMENT BY THE SPDM WG

SPDM over TCP/IP Binding

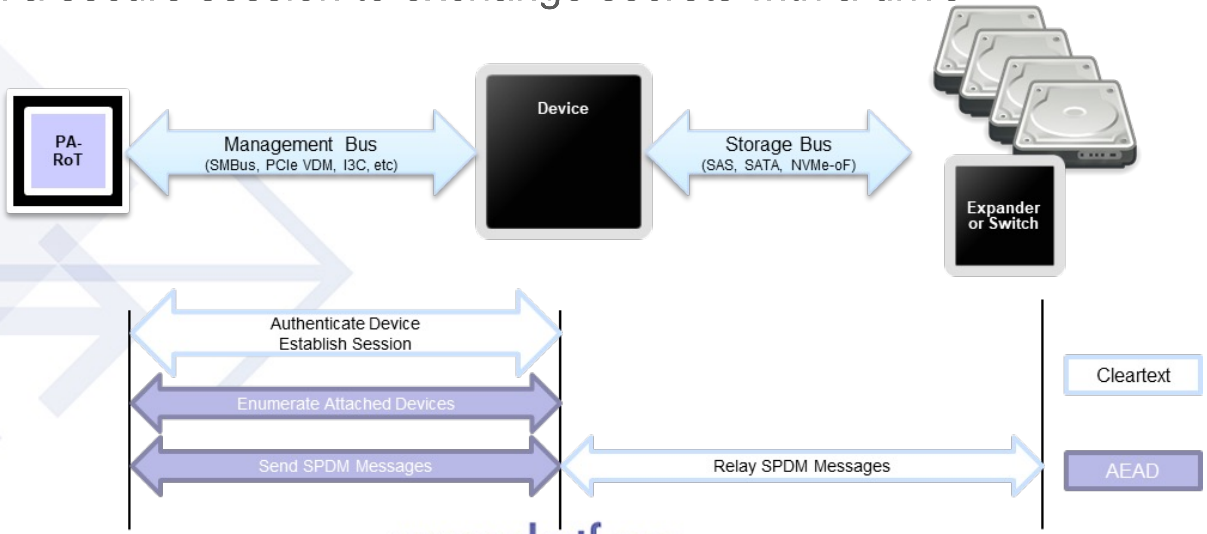
- Adds a binding spec (DSP0287) to support SPDM over TCP/IP connections
  - Target is Ethernet-based fabrics and use in conjunction with RDMA
  - Provides standardized way to establish the security of a TCP/IP fabric before opening a memory window
  - Reuses existing SPDM protocol (DSP0274)
- Sample use cases
  - Provisioning certificates from a CA server to a device
  - Using a remote Attester to offload SPDM attestation from the local PA-RoT
  - Allowing an SPDM Requester to retrieve reference measurements
  - Secure data transfer between a device and server that is lighter than HTTP/TLS





SPDM over Storage Binding

- Adds a binding spec (DSP0286) to support SPDM over storage transports
  - Supports SAS, SATA, and NVMe over fabrics
  - Leverages existing storage commands (IF-SEND and IF-RECV)
  - Reuses existing SPDM protocol (DSP0274)
- Sample use cases
  - Authenticate a drive before using it for data storage
  - Attest the state of a drive
  - Establish a secure session to exchange secrets with a drive



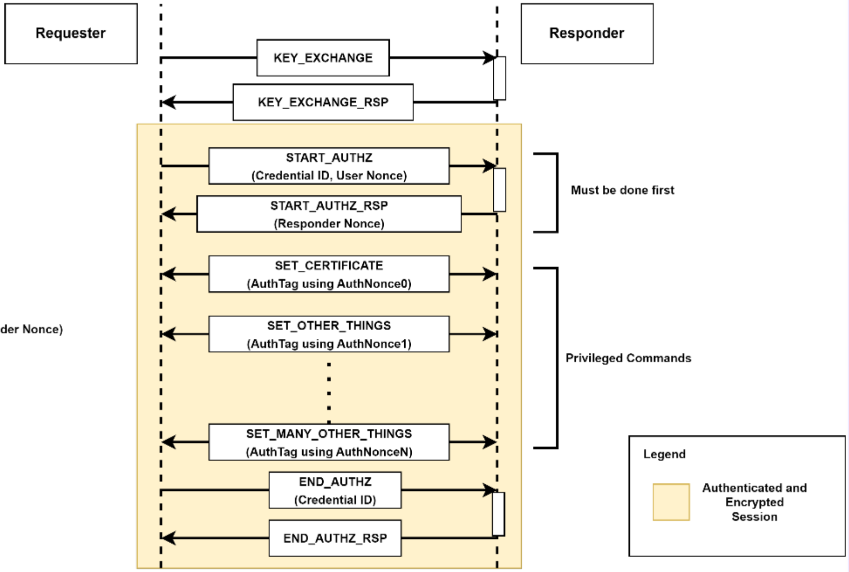
Authorization Specification

- Creates a new specification to enable authorization
  - Provides a mechanism to determine whether the requesting entity has the correct privileges to perform a protected action
- Generalized approach
  - Leverages SPDM Secured Messages
  - Can be used for SPDM, PLDM, vendor-defined messages and more
- Supports multiple sets of privileges
- Scalable to support large numbers of endpoints (data centers)

AuthNonce

AuthNonce = (User Nonce || Responder Nonce)

AuthNonce0 = AuthNonce
AuthNonce1 = AuthNonce0 + 1
...
...
...
AuthNonceN = AuthNonce(N-1) + 1



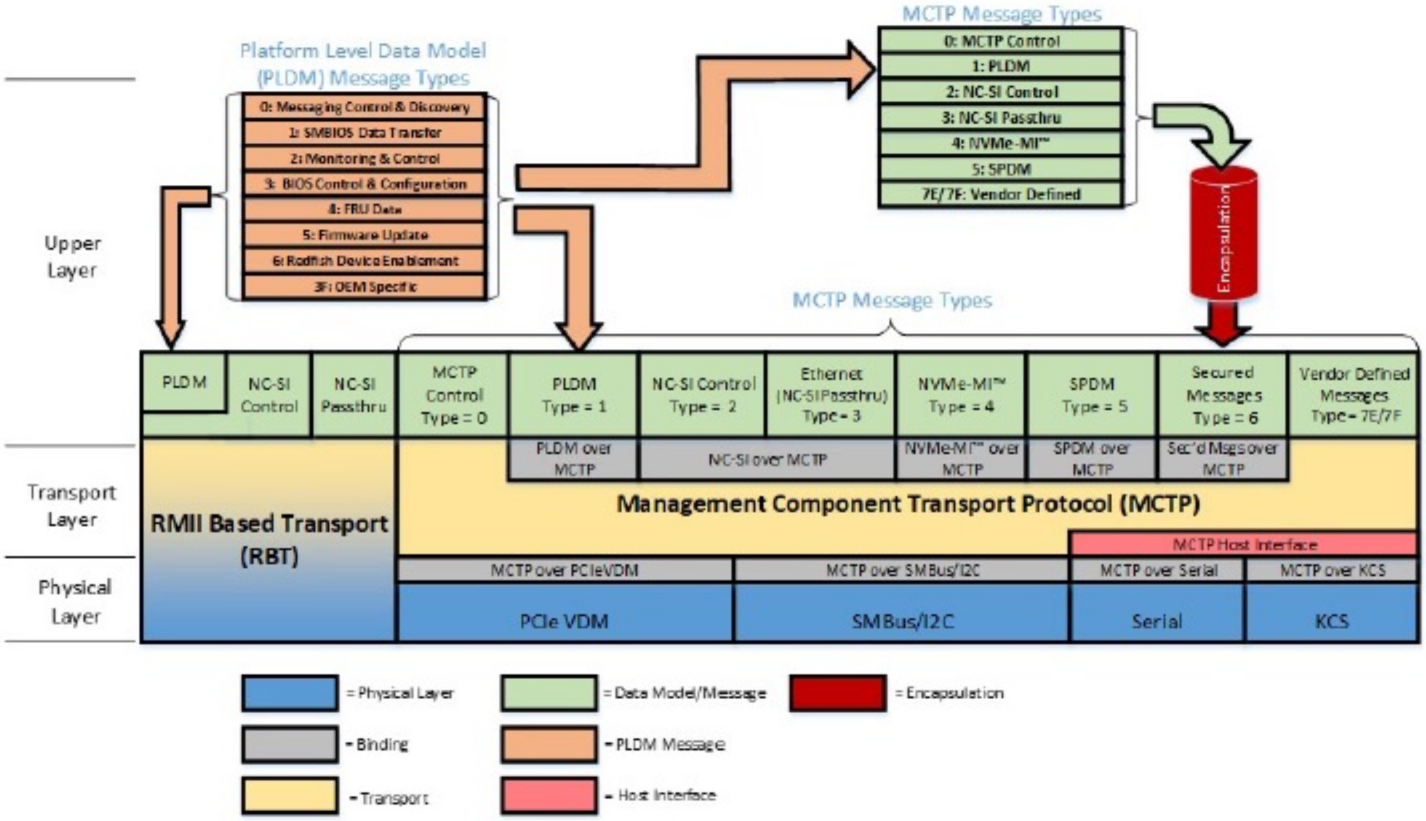
Take Aways

- SPDM protocol is a prominent industry standard for Component and Device Attestation
- Has traction among other industry standard organizations and among component and system vendors
  - DMTF plans to submit the SPDM specification to ISO for ratification
- Use cases and specification work are expanding
- DMTF seeks participation, collaboration and input from the industry

Backup

SPDM over MCTP including Encrypted Messages

Figure 2 — SPDM over MCTP



Platform Level Data Model (PLDM) Message Types

0: Messaging Control & Discovery
1: SMBIOS Data Transfer
2: Monitoring & Control
3: BIOS Control & Configuration
4: FRU Data
5: Firmware Update
6: Redfish Device Enablement
3F: OEM Specific

MCTP Message Types

0: MCTP Control
1: PLDM
2: NC-SI Control
3: NC-SI Passthru
4: NVMe-MI™
5: SPDM
7E/7F: Vendor Defined

Encapsulation

Upper Layer

Transport Layer

Physical Layer

MCTP Message Types

PLDM
NC-SI Control
NC-SI Passthru
MCTP Control Type = 0
PLDM Type = 1
NC-SI Control Type = 2
Ethernet (NC-SI Passthru) Type = 3
NVMe-MI™ Type = 4
SPDM Type = 5
Secured Messages Type = 6
Vendor Defined Messages Type = 7E/7F

PLDM over MCTP
NC-SI over MCTP
NVMe-MI™ over MCTP
SPDM over MCTP
Sec'd Msgs over MCTP

RMII Based Transport (RBT)

Management Component Transport Protocol (MCTP)

MCTP Host Interface

MCTP over PCIeVDM
MCTP over SMBus/I2C
MCTP over Serial
MCTP over KCS

PCIe VDM
SMBus/I2C
Serial
KCS

🟦 = Physical Layer
🟩 = Data Model/Message
🟥 = Encapsulation
⬜ = Binding
🟧 = PLDM Message
🟨 = Transport
🟥 = Host Interface