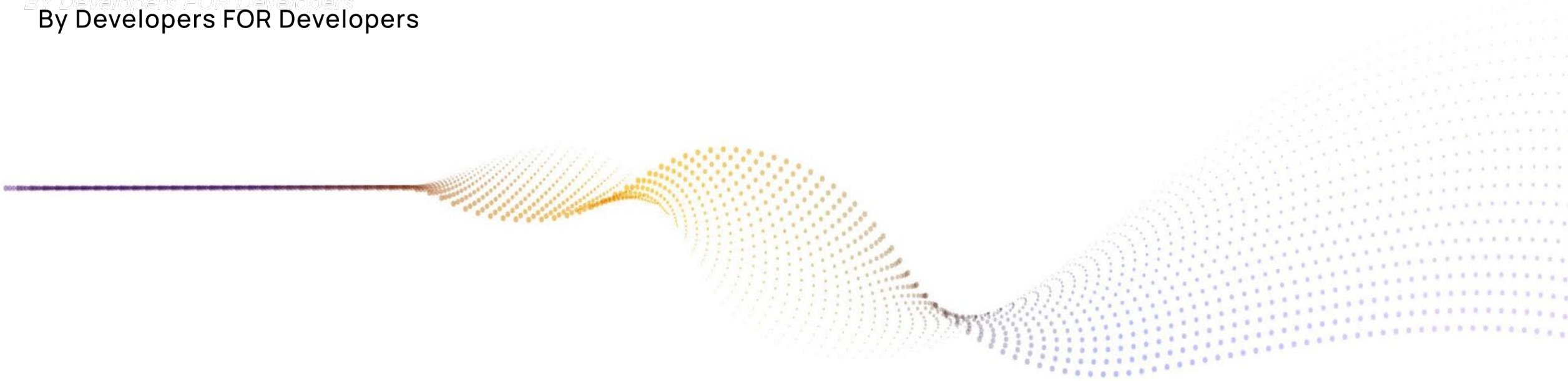
SNIA DEVELOPER CONFERENCE

SDC 25

By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025



SPDM

Post-Quantum, Authorization, and Storage

www.sniadeveloper.org

Disclaimer

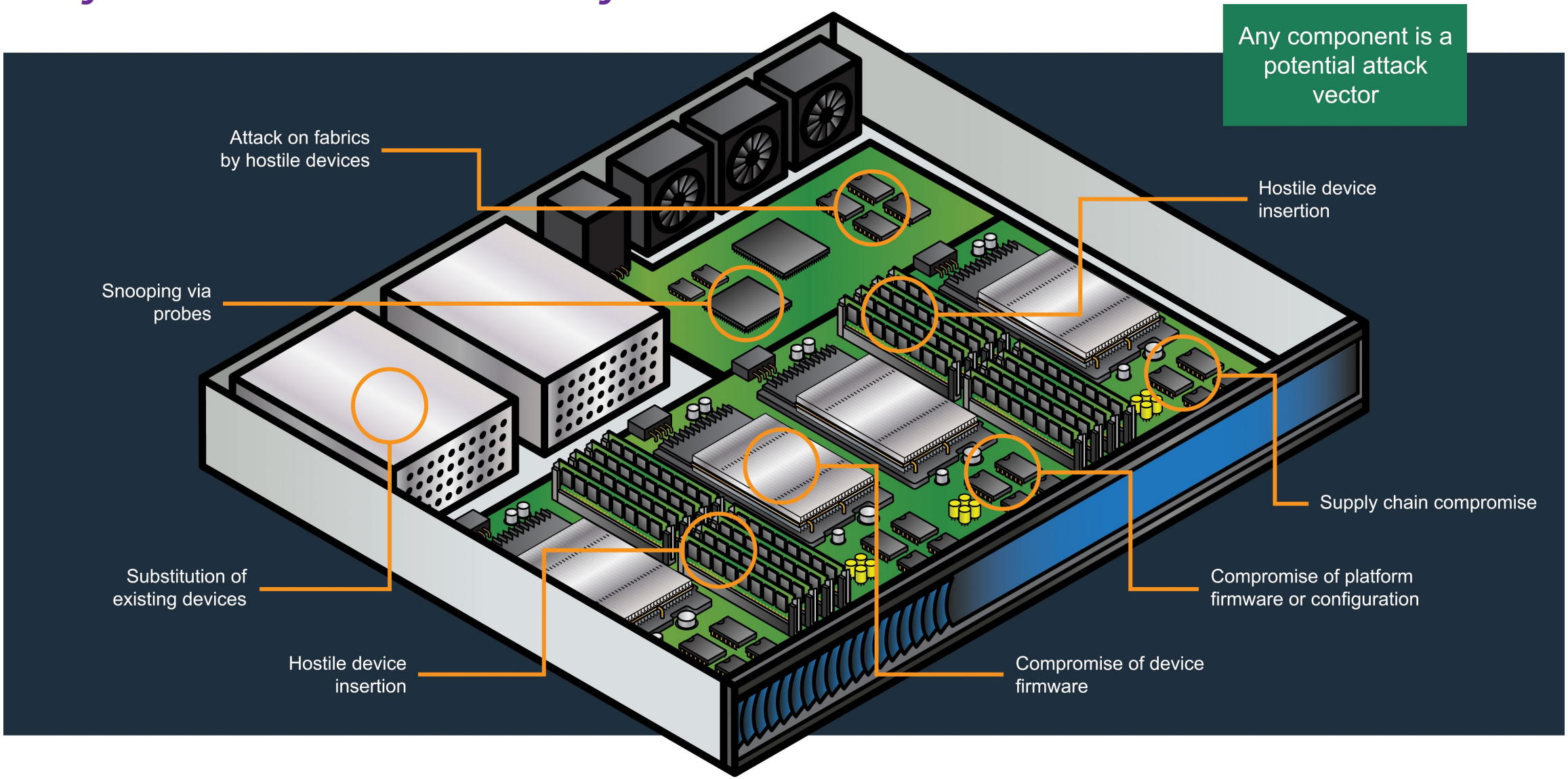
- The information in this presentation represents a snapshot of work in progress within the DMTF SPDM WG.
- This information is subject to change without notice. The standard specifications remain the normative reference for all information.
- For additional information, see the DMTF website.
- This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

Alliance Partners and Adopters

Agenda

- SPDM Background
- Update on post-quantum cryptography and SPDM v1.4
- Update on SPDM to Storage Binding
- SPDM Authorization

Why Platform Security



Why SPDM

- The industry needs a common solution that works for the control plane and data plane regardless of technology or protocol and integrates into the solutions being proposed by other open communities. This decreases costs while increasing security.

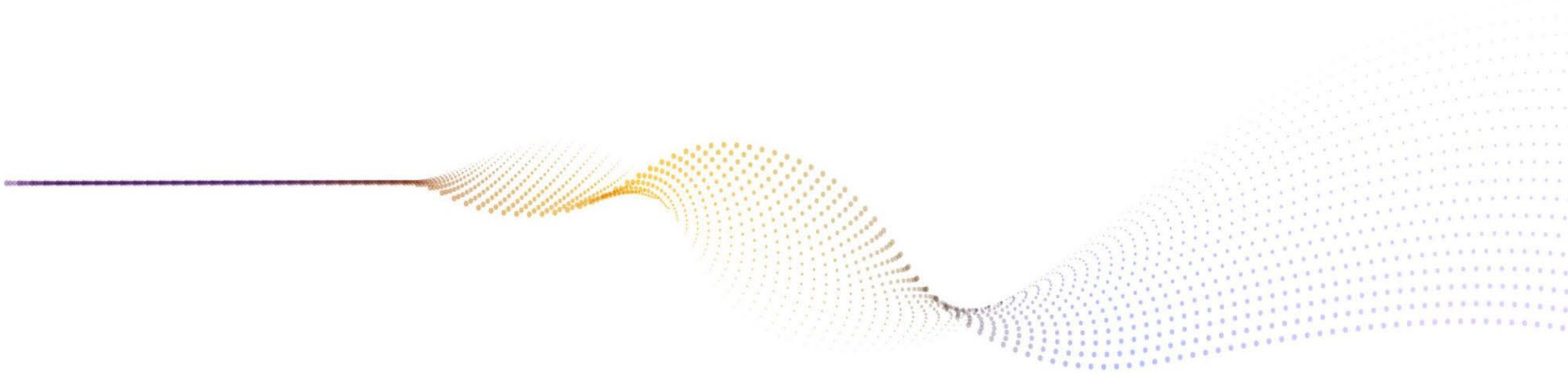
- Having an open-source code base (DMTF's libspdm) that can be leveraged by both ends of the wire (Requester and Responder in SPDM language) helps seed the industry and allows researchers to validate the security of the standard.

SPDM's Overall Goals

- All SPDM features fall into at least one of these main goals:
  - Device Attestation and Authentication
  - Secure Communication over any transport
- 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.)
- Support latest cryptography standards
  - Especially post quantum crypto (PQC) algorithms, such as ML-DSA, ML-KEM, and SLH-DSA

SPDM Feature Summary

- Version 1.0:
  - Measurements
  - Device Attestation and Authentication
- Version 1.1:
  - Secure Session
  - Public Key Exchange
  - Symmetric Key Exchange
  - Mutual Authentication
- Version 1.2:
  - 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
- Version 1.3
  - Eventing
  - Multi-Key
  - Generic Certificates
  - MEL & HEM
  - Endpoint Info
- Bindings
  - MCTP to SPDM
  - MCTP to Secure Messages
  - Secure Messages to SPDM
  - SPDM over TCP
  - SPDM over Storage (NVMe, SAS, SATA)
- Other Security Specs
  - Authorization



SPDM 1.4

Post-Quantum Cryptography

SPDM 1.4 Feature Additions

- PQC support in algorithm negotiation, certificates, and key pair information messages
  - ML-KEM (FIPS 203), ML-DSA (FIPS 204), SLH-DSA (FIPS 205)
- Certificate slot management
  - Defines “banks” of certificate slots, for managing certificates of different signing algorithms.
  - A bank contains up to 8 slots of certificates of the same signing algorithm.
  - A device may support multiple signing algorithms, hence multiple banks.
- Miscellaneous:
  - Added SET_KEY_PAIR_RESET_CAP
  - Added salt length requirement for RSA-PSS schemes.
  - Minor clarifications and fixes

Post Quantum Algorithm Support

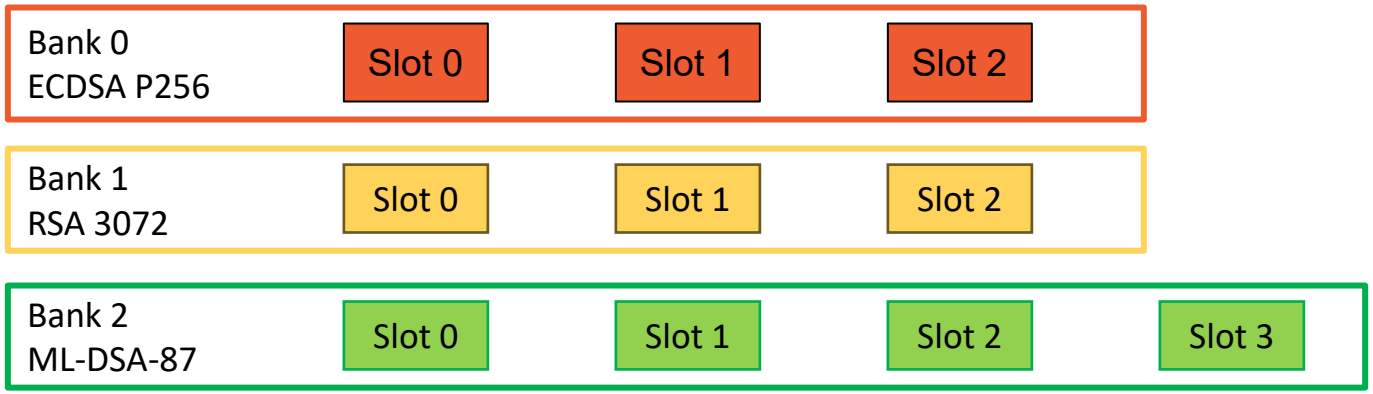
- Supports following standards
  - FIPS 203 ML-KEM for key encapsulation, all security categories
  - FIPS 204 ML-DSA for digital signature, all security categories
  - FIPS 205 SLH-DSA for digital signature, all parameters sets
  - NIST SP 800-227 Recommendations for Key-Encapsulation Mechanisms
- No hybrid support in SPDM 1.4.
  - NEGOTIATE_ALGORITHMS/ALGORITHMS cannot negotiate 2 or more algorithms (e.g., a traditional algorithm and a PQC algorithm) for a given category.

SPDM Messages Impacted by PQC

Message	Added / expanded fields for PQC	Digital Signature – added ML-DSA and SLH-DSA	Key Exchange – added ML-KEM
Core Messages			
CERTIFICATES	Yes		
CHALLENGE_AUTH		Yes	
ENDPOINT_INFO		Yes	
MEASUREMENTS		Yes	
KEY_EXCHANGE / _RSP	Yes	Yes	Yes
FINISH		Yes	
CSR		Yes	
SLOT_MANAGEMENT / _RESP	Yes		
Supporting Messages			
NEGOTIATE_ / ALGORITHMS	Yes		
SET_CERTIFICATE	Yes		
SET_KEY_PAIR_INFO	Yes		
KEY_PAIR_INFO	Yes		

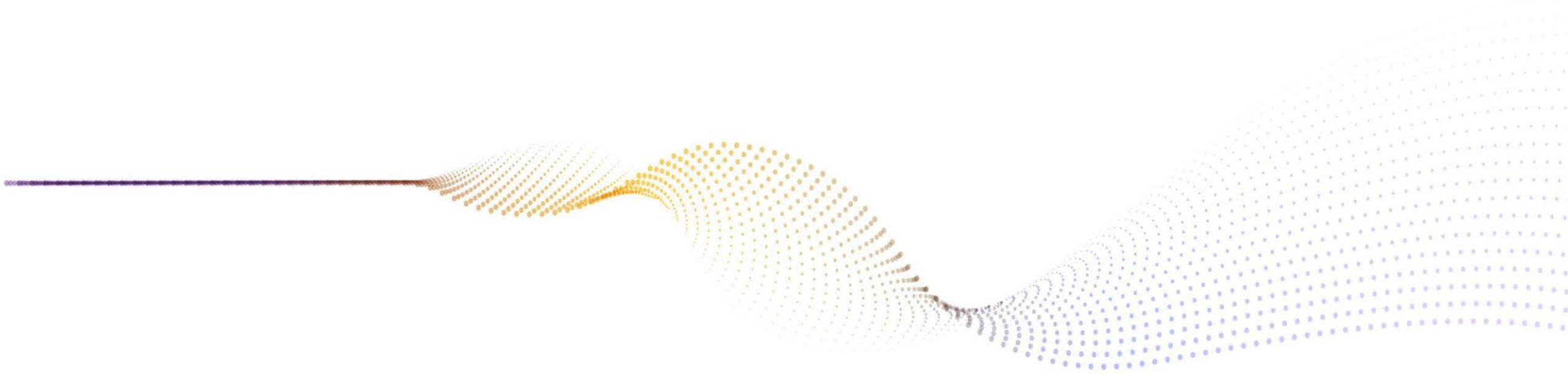
Certificate Slot Management With Bank

- Today, many devices support only one signing algorithm
  - The device can have up to 8 slots (numbered 0-7) that store certificate chains associated with the supported signing algorithm.
- When multiple signing algorithms are supported by the device, each algorithm can support up to 8 slots.
  - The bank of slots is implicitly selected via NEGOTIATE_ALGORITHMS / ALGORITHMS, providing no way to manage other banks of certificates.
- SPDM 1.4 provides commands to manage certificate chains
  - Every signing algorithm is allocated a bank.
  - The selected signing algorithm does not affect the new certificate slot management commands.



References

- SPDM 1.4.0
  - https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.4.0.pdf
- FIPS 203
  - https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf
- FIPS 204
  - https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf
- FIPS 205
  - https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf
- SP 800-227 Initial Public Draft
  - https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-227.ipd.pdf



SPDM to Storage Binding

Background

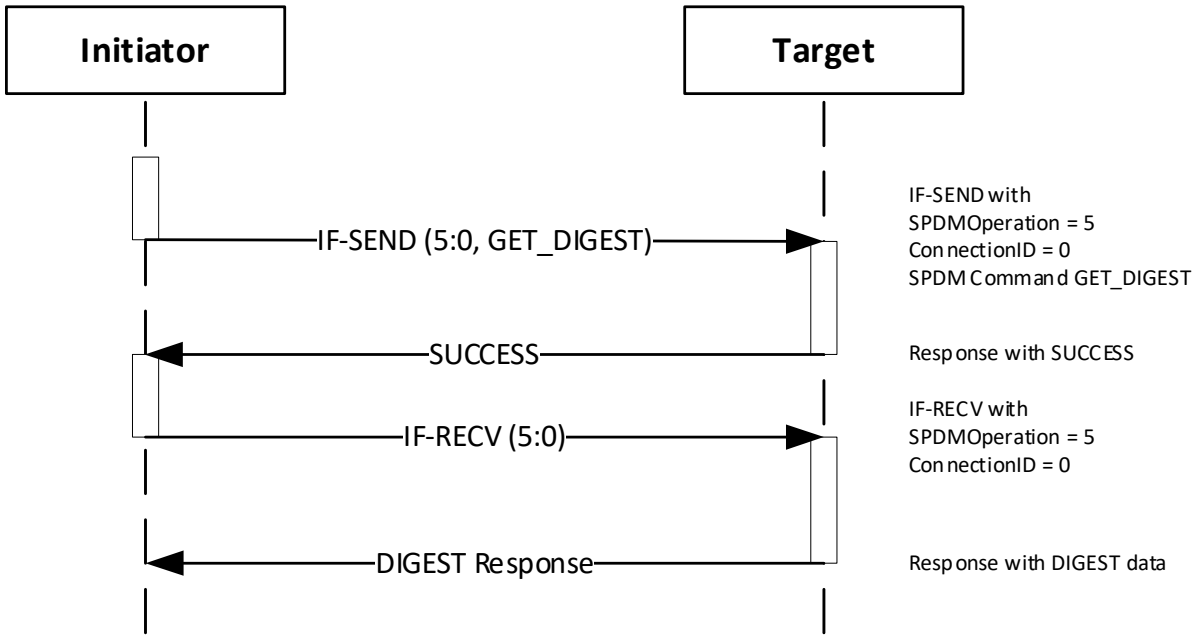
- Many NVMe® devices support MCTP transports and can use SPDM
  - These transports are not available for SAS or SATA
  - Often are not available for NVMe over Fabric attached controllers
- There is a desire to extend SPDM attestation and security capabilities to storage devices
  - In many systems, storage devices are a high proportion of the system device content
  - There is a desire to consistently manage all storage devices
- Extending SPDM to storage transports enables existing SPDM software stacks to work with a broader device set
- The binding is for SPDM, not for generic MCTP

Storage Binding Status

- Version 1.0.0 published in May 2025
  - DSP0286 - https://www.dmtf.org/dsp/DSP0286
- Supports all current SPDM command sets
  - Core SPDM functionality (DSP0274) versions 1.0 through 1.4
    - Includes support for PQC
  - Supports secured messages (DSP0277) versions 1.x
  - Will support SPDM Authorization (DSP0289) when published

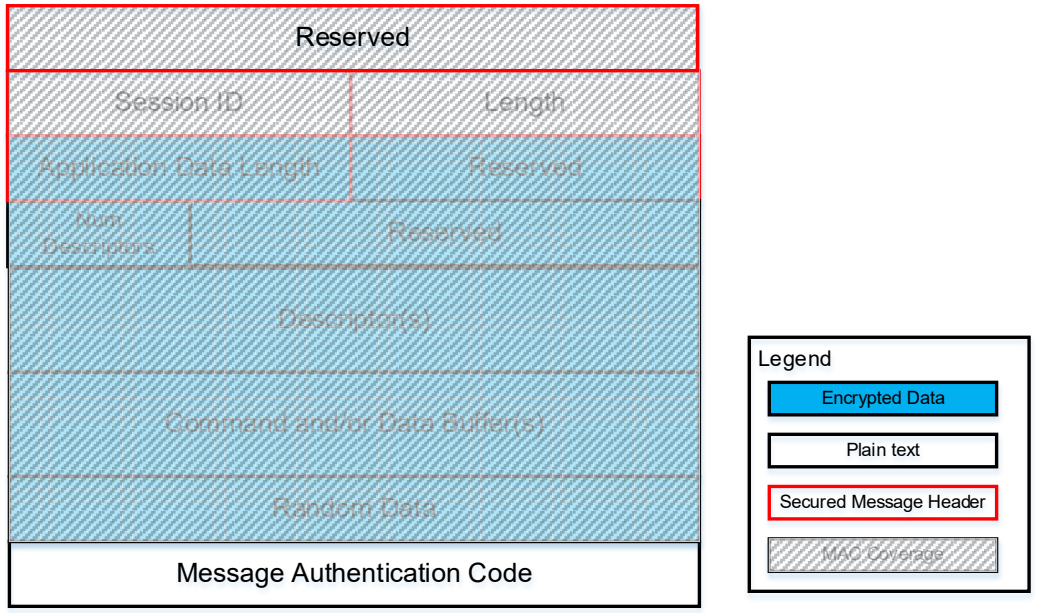
SPDM Command Flow

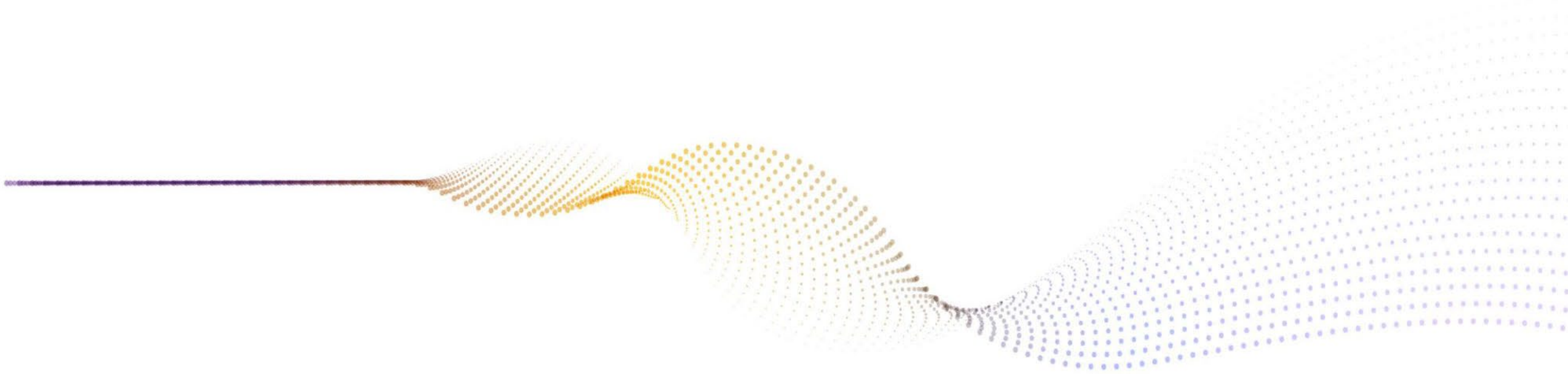
- SPDM commands (DSP0274) are exchanged using the mandatory SPDM Storage Message operation
  - Equivalent of MCTP type 5 commands
- The request is sent using IF-SEND
- Response is retrieved using a separate IF-RECV



SPDM Secured Messages Format

- Transmits in the data buffer of an IF-SEND or IF-RECV
- Carries messages from DSP0277
  - Not intended for disk IO
- The data section starts with a list of descriptors
  - One descriptor describes one buffer
    - Type, offset, length
  - Buffers can be a command, data, or response
  - There are rules on mixing
    - Such as only one command per message
    - Response must follow the protocol for the request
    - Request/response rules still apply





SPDM Authorization

Authorization

- Definition:
  - Determining if the requesting entity has the appropriate privileges to perform protected actions. If yes, to allow them to perform those protected actions.
- Scope:
  - Provide a general mechanism for any use case (e.g., SPDM, PLDM, other present and future PMCI WG use cases, alliance partners, industry) to perform authorization.
    - Examples:
      - PLDM FW Update
      - PLDM Type 2 (ex. Configuring event receiver, configuring sensor thresholds)
      - PLDM Type 6 (get vs patch/post i.e. RDE)
      - SPDM Set Certs (and other future “set” commands).

Assumptions

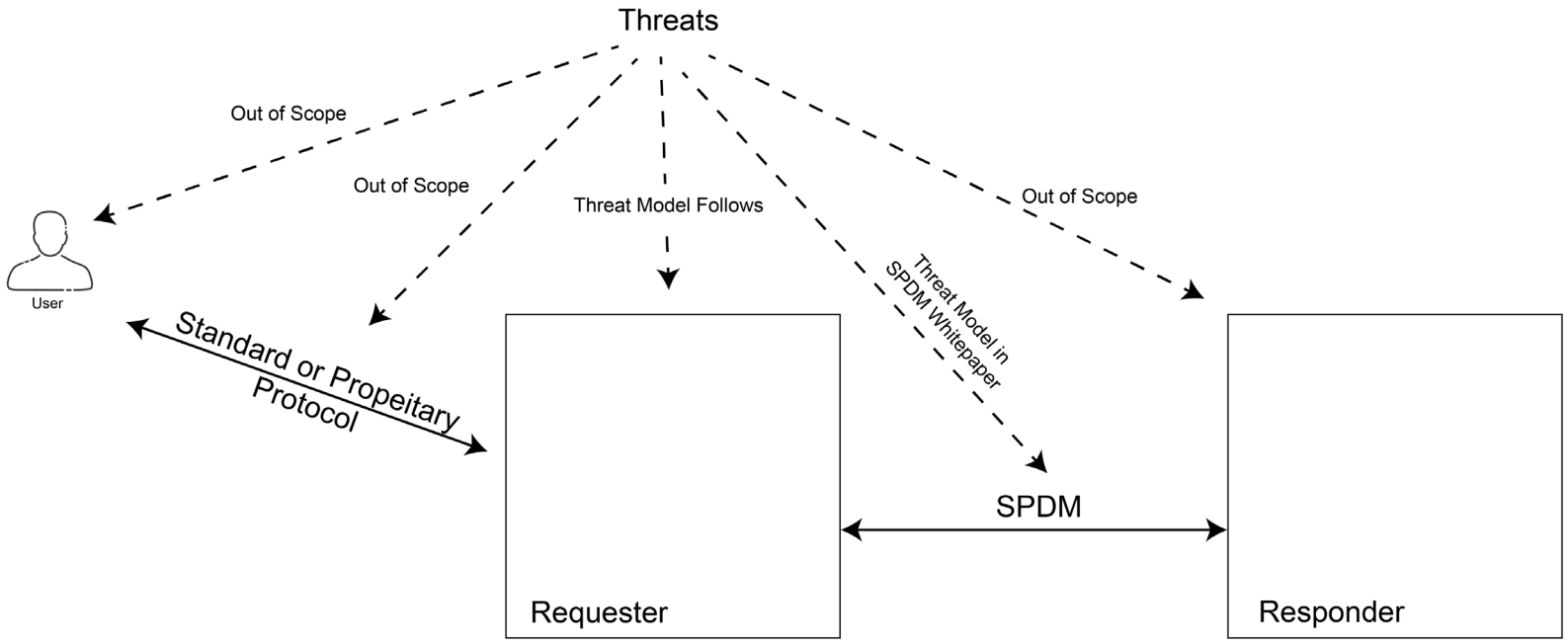
- The endpoints in discussion communicate using SPDM (DSP0274) and SPDM Secured Messages (DSP0277)
  - Communication can use any transport that supports the above commands, including the SPDM to storage binding
- To bootstrap Authorization, there needs to be a provisioning step for initial credential
- Definition of Policy profiles is out of scope for the Authorization specification

High Level Architectural Components

- Authorization Flow
  - Use SPDM Sessions between Requester/Responder pair (simplifies supported options, baseline security)
  - Specify how to authorize generic messages
- Credential and Policy Management
  - Types of Credentials
    - Asymmetric Key Pair (Focus of initial release)
  - Credential and Credential Policy
    - Standardize provisioning of credentials and associating them with their authorization policy
    - Authorization policy itself should be specified by the protocol leveraging this authorization specification

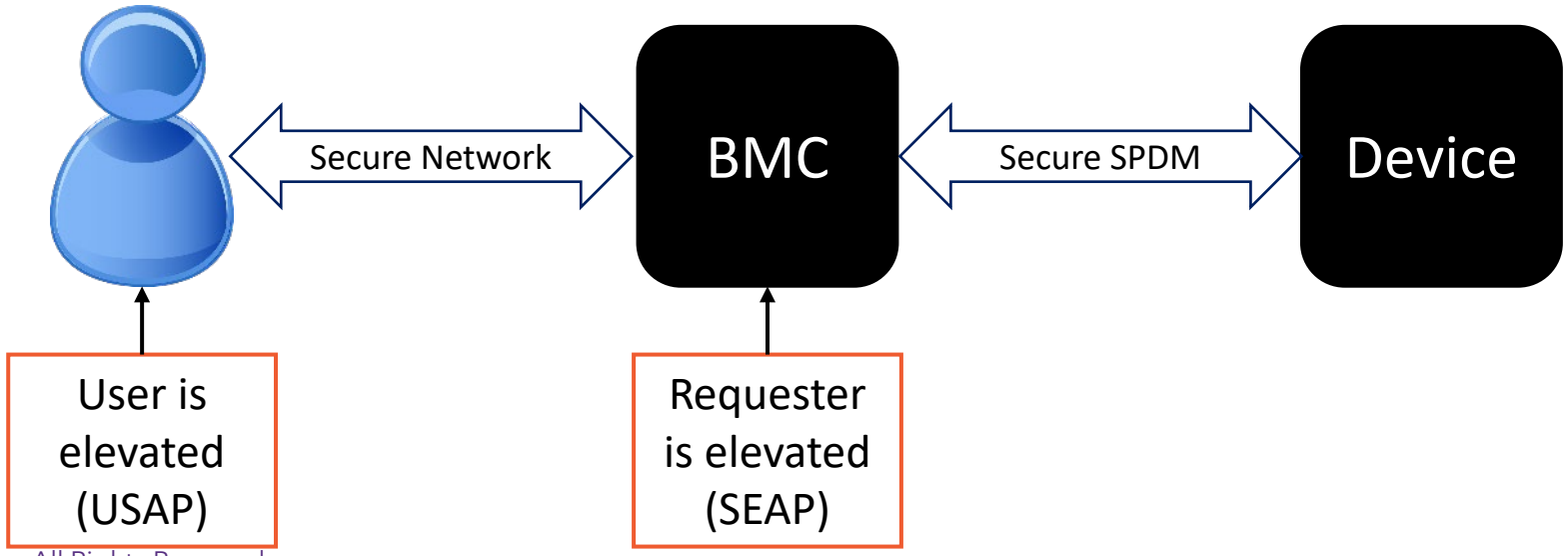
Authorization Threat Model Overview

- 2 Use Cases – the User is the authorized actor
  - SPDM Requester represents user (SEAP)
  - SPDM Requester is a proxy for user (USAP)



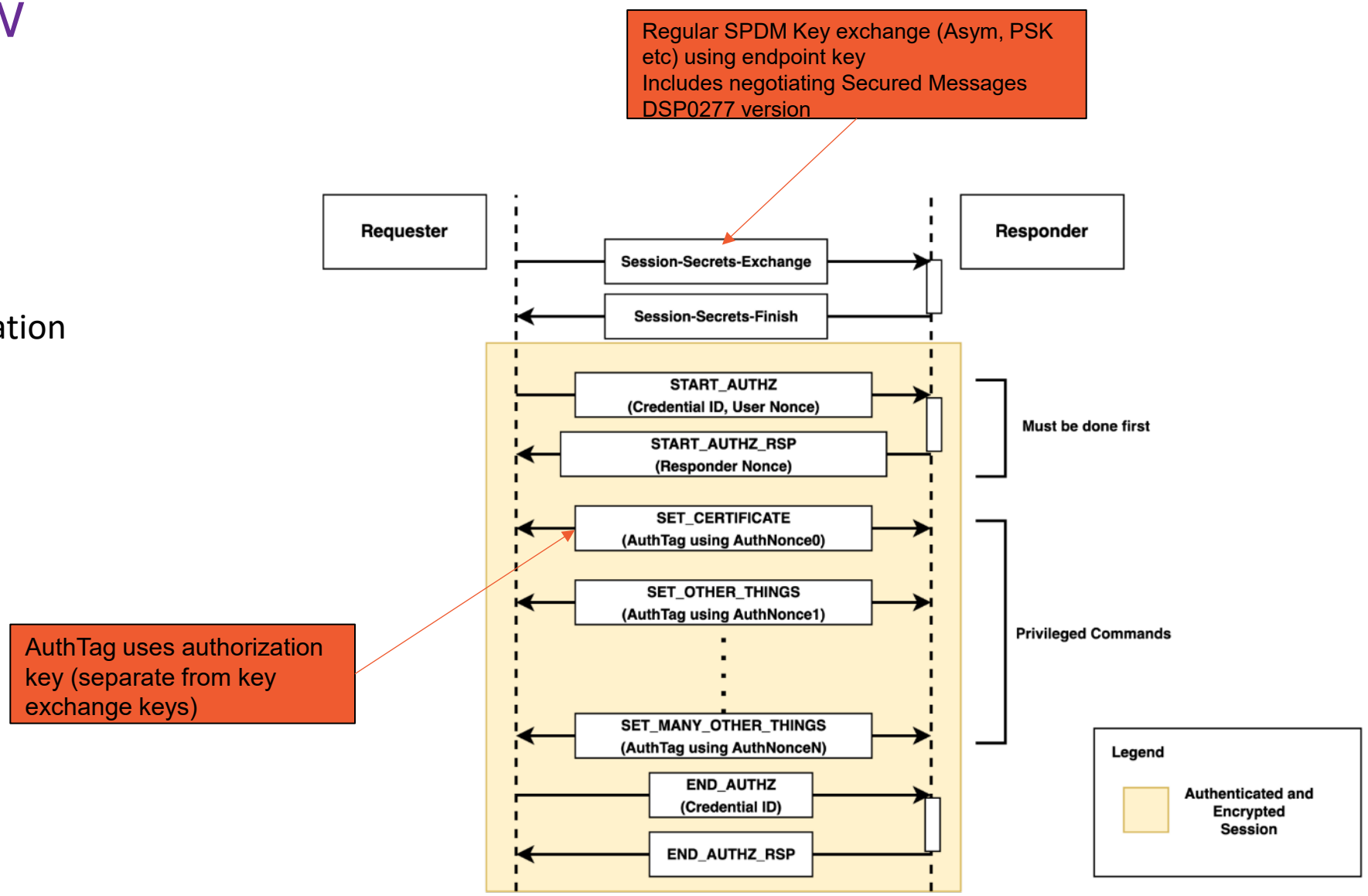
SEAP vs. USAP Example

- USAP
  - The user elevates privilege
  - Selective commands from that user are elevated
  - Like an SSH connection
- SEAP
  - The initiator elevates privilege
  - All commands from that initiator are elevated
  - Like Linux ‘sudo’ command



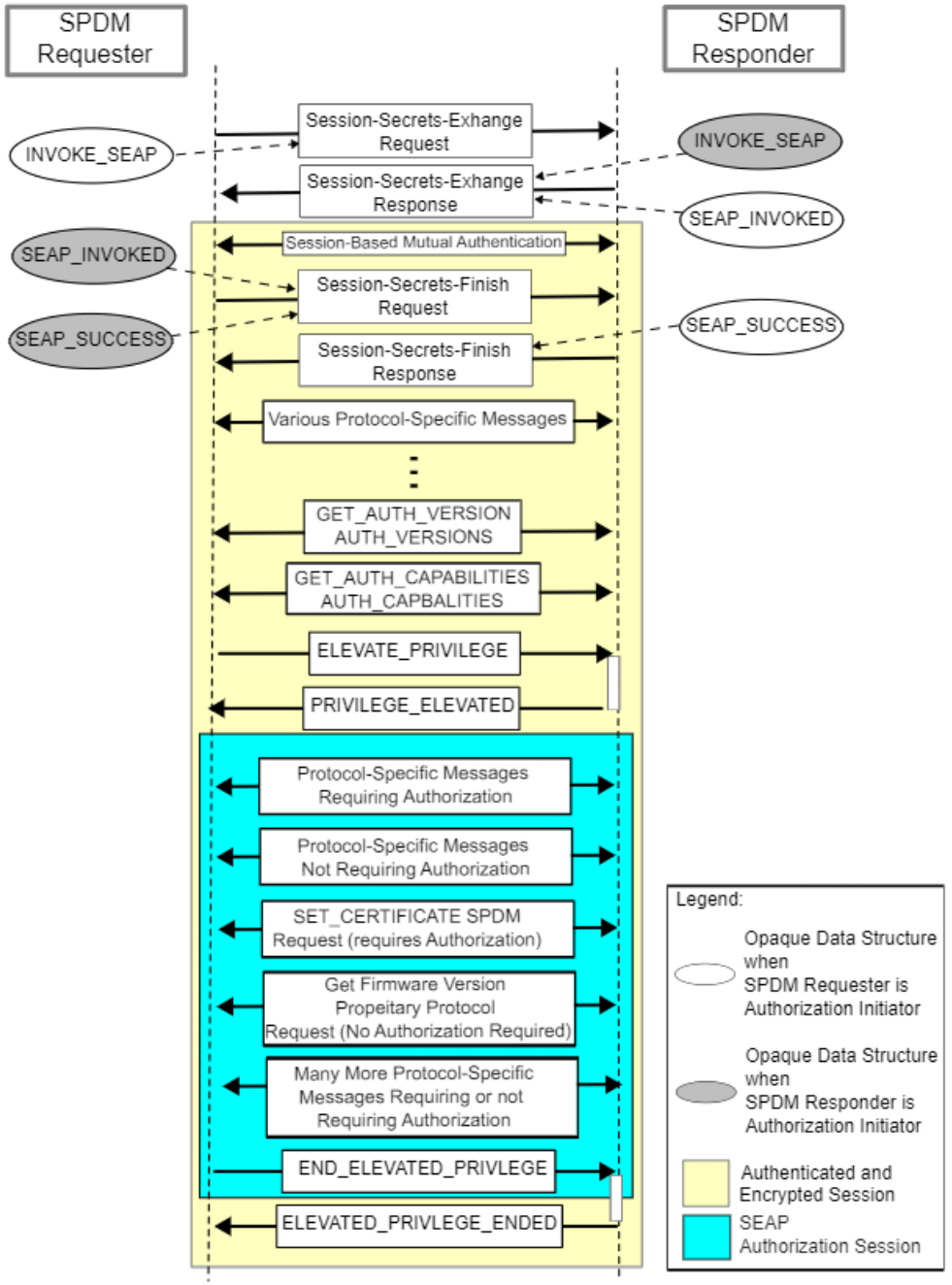
USAP Flow

User-Specific Authorization Process



SEAP Flow

SPDM Endpoint Authorization Process



Credential Provisioning

- DSP0289 defines 8 persistent Credential Slots, minimum supported is one
  - The ability to clear slot contents will be included in v1.0
- Slot 0 – Only provisioned in a trusted environment (ex: Secure Manufacturing, secure onboarding)
  - Bootstrap authorization
- All other slots can be provisioned in
  - A trusted environment
  - Authorized via a selected credential(s) already provisioned
- All credentials associated with a policy
  - Defines what the credential can be used to authorize (ex: SET_CERTIFICATE, PLDM FW Activation etc)
- DSP0289 defines credential provisioning commands for credentials and policies

Policy

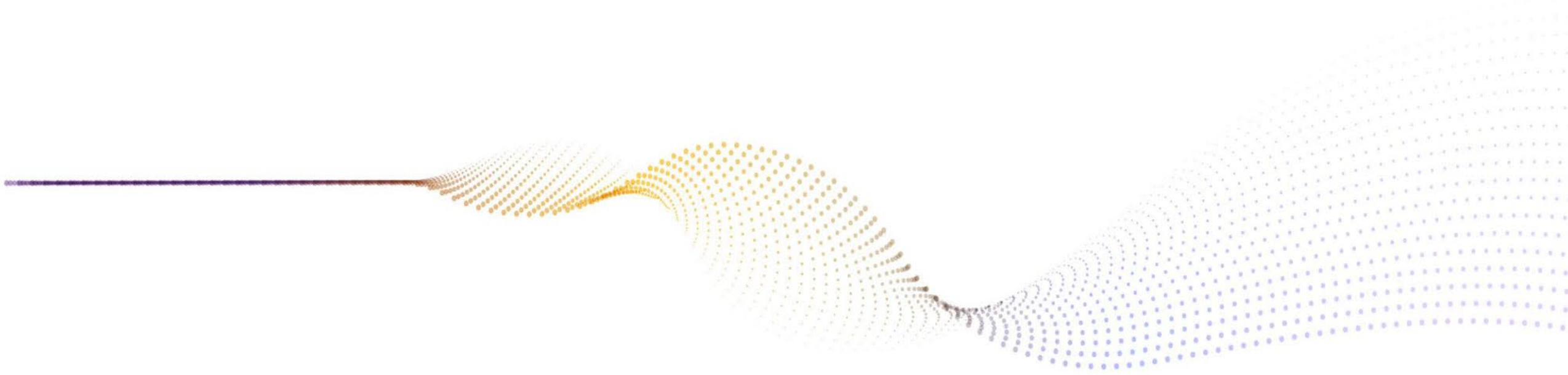
- A policy defines an Authorization Initiator's access to one or more protected resources
  - For instance, can an Authorization Initiator update firmware?
- Authorization Spec gives a policy framework
  - Discovery and assignment of policies
- Policies are not defined in the Authorization Spec
  - The Authorization Spec only defines policies related to its own commands
  - Does not define SPDM (DSP0274) policies
  - Does not define policies for any other standards body

Next Steps for Policies

- Authorization will require policies to be defined before it can be used
- The SPDM Working Group will define policies for SPDM (DSP0274)
  - This will be a template that can be a starting point for others
- We need partner organizations to define policies
  - For example, the SPDM people do not know how to correctly define policies for NVMe-MI
- Vendors can also define policies
  - Helps the vendor build an authorization framework around a private API

Call to Action

- Start your SPDM post-quantum cryptography migration now
- Understand the SPDM to Storage Binding and how you can use it in your environment
- Engage with the Authorization Specification
  - Understand how you want to use it in your environments
  - Start working on authorization policies
  - Incorporate authorization flows and policies into the OCP Attestation Spec
- Call for SPDM version 1.5 features
  - Feedback Portal - https://www.dmtf.org/standards/feedback



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

