SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

SPDM & Post Quantum Crypto

SPDM WG

Presented by Jeff Hilland Distinguished Technologist HPE; President DMTF and
Brett Henning Security Architect Broadcom

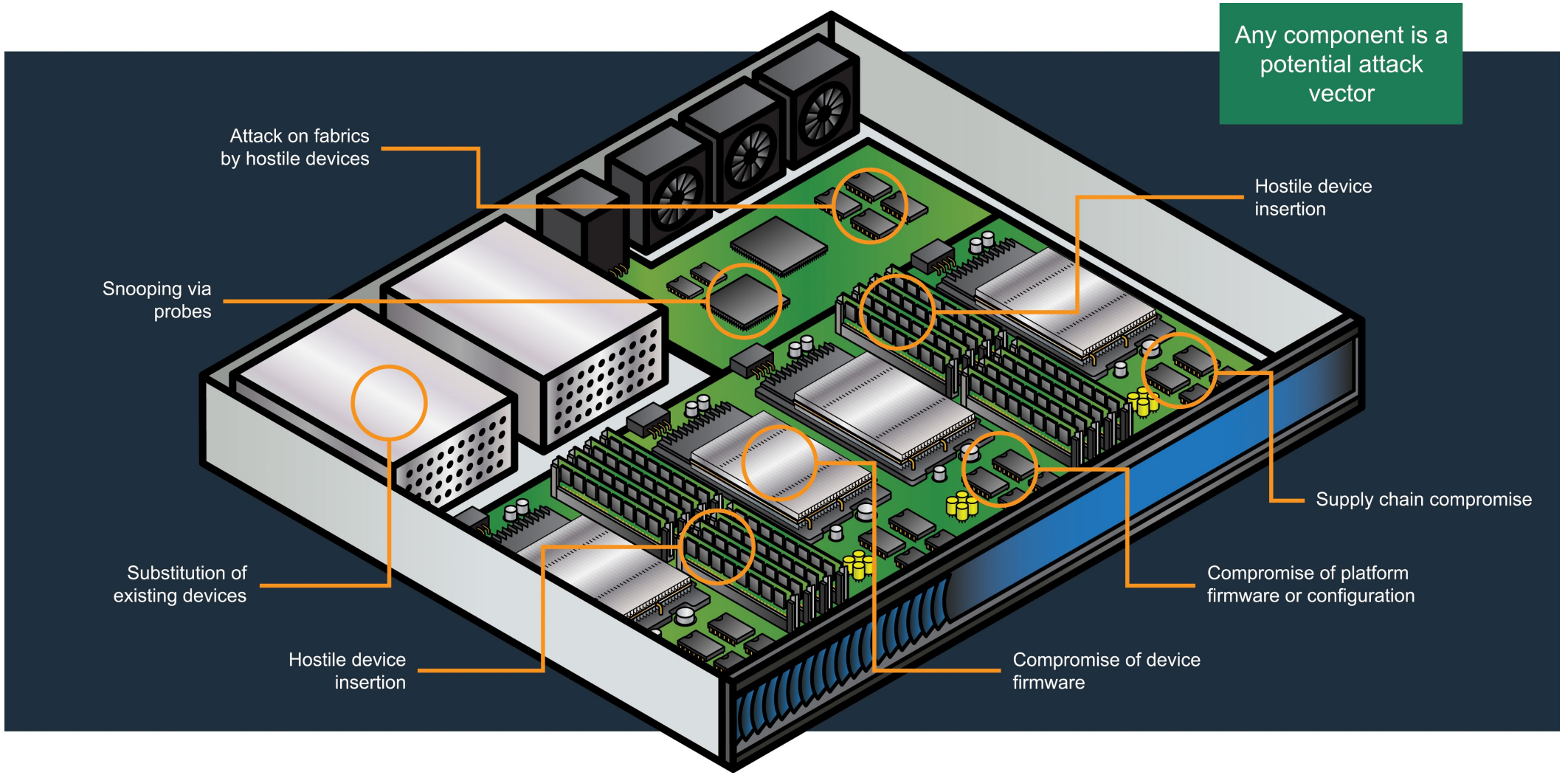
Disclaimer

- 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.

Agenda

- SPDM Background
- SPDM for Storage Binding
- SPDM PQC Update

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.)

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)

SPDM for Storage

Motivation

- 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 also 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

Scope of Work

- Enable a binding for SPDM messages over SAS (SCSI), SATA (ATA), and NVMe
  - The new standard will be DSP0286
- The binding is for SPDM, not for generic MCTP
  - The Security Protocol ID that is assigned for SPDM is 0xE8
- Both SPDM messages and SPDM Secured Messages are supported
  - For those familiar with SPDM, think of this spec as covering both DSP0275 and DSP0276
  - Goal is to support new SPDM revisions without major changes to the binding specification

SPDM Storage Operations

- The binding spec defines the use of the SECURITY PROTOCOL SPECIFIC field in the IF-SEND and IF-RECV commands
  - This is a 16-bit field
  - Used to manage the binding, and not directly for SPDM

Byte	Bits	Field Name	Description
0	1:0	ConnectionID	Manage the connection in use. A connection defines a pairing of messages, such as Connection 0 tracks a request with a matching response. At a high-level, each connection is an independent SPDM state machine. At least one connection (ConnectionID = 0) shall be supported.
0	7:2	SPDMOperation	The command code for the operation.
1	7:0	Reserved	

Binding Discovery

- SPDM Storage Discovery is a mandatory command and used to determine if a device supports the SPDM binding spec and all details
  - Modeled on SCSI INQUIRY
  - Intended to be the first SPDM related command sent
    - The initiator may cache this info, so there is no need for a device to enforce that this is the first command
- The initiator reads the buffer from the device
  - If the SPDM storage binding is not supported, the device returns Invalid Field in CDB

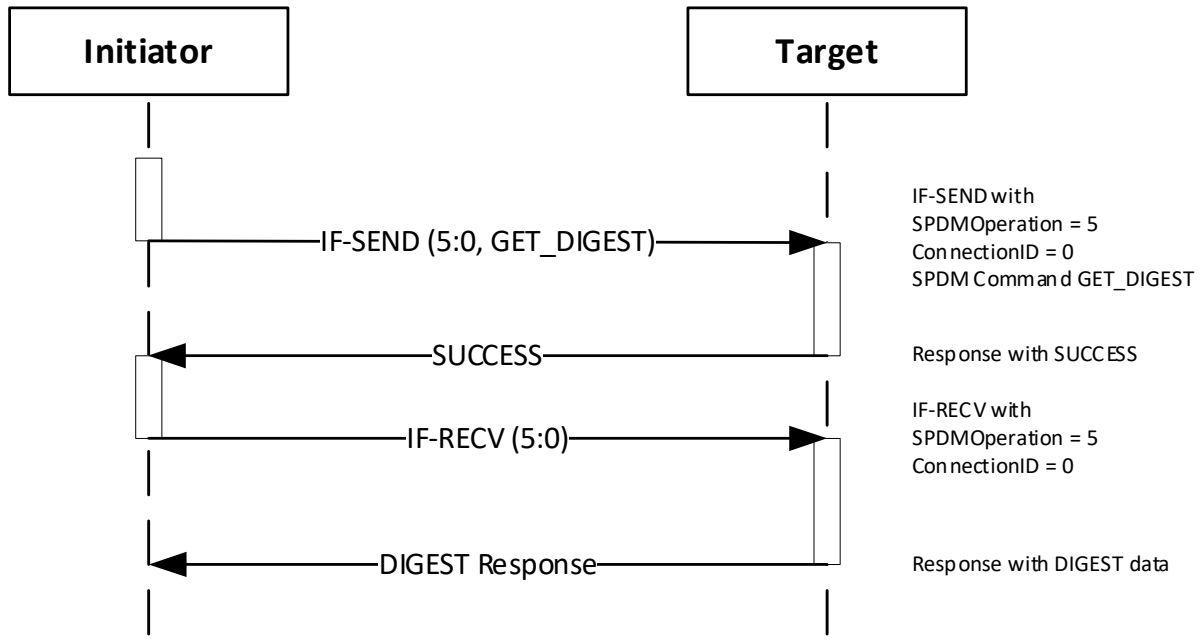
Field	Details
DataLength	Returns the number of available bytes (may be larger than the allocation buffer)
Version	Version of the binding spec that the device supports
MaxConnectionID	The highest supported ConnectionID, the device must support at least 1 (i.e. MaxConnectionID of 0)
SupportedOperations	Bitmap that indicates which SPDMOperation codes are supported

Use of IF-SEND and IF-RECV

- All in-scope storage specs support a Security Protocol In and Out command
  - These are generically referred to as IF-SEND (host to device) and IF-RECV (device to host)
- SPDM defines all commands as a request and response pair
  - Storage specs would call this a bidirectional transfer, which is not commonly supported
- The generalized approach is to use an IF-SEND for the request, followed by an IF-RECV to fetch the response
  - There will be further discussion on how the send and receive are paired
  - Sends from the device to the host are supported using the encapsulated flow (see subsequent slides)

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



Pending Info Problem

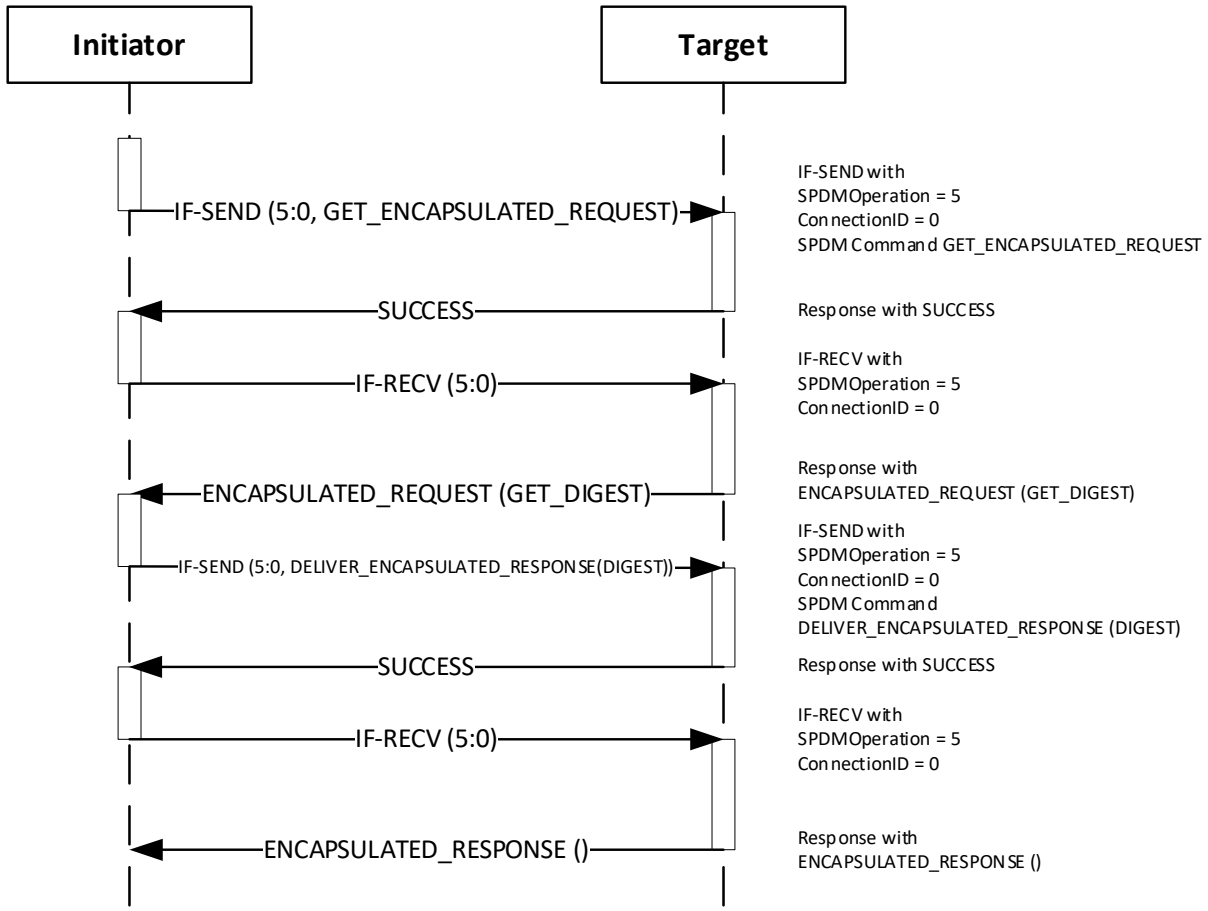
- Initiator needs to know how much data is waiting for the IF-RECV?
  - Allocate a max sized buffer for every command (64 KB or chunk size)
  - Have a lookup table to project the expected size for each command
  - Send SPDM Storage Pending Info
- SPDM Storage Pending Info returns details about any pending response
  - Considered transport level and not included in any transcript hashes

Field	Details
DataLength	Returns the number of available bytes (may be larger than the allocation buffer)
Version	Version of the binding spec that the device is using
PendingInfoFlags	Flags about the response info: Bit 0 – ValidResponse pending
ResponseLength	Length of pending response data, in bytes, for this connection

Encapsulated Flow

- Devices cannot start a transaction, but may want to support SPDM events or mutual authentication
  - SPDM has built-in support for initiating commands from a Responder to a Requester on unidirectional busses
    - Encapsulated Request
- The initiator asks the device (IF-SEND) if it has any commands to send, the reply from the device (IF-RECV) contains the request or “None”
- Inefficient, but it works and uses existing structures
- See next slide for diagram

Encapsulated Flow Diagram



Transcript Hash

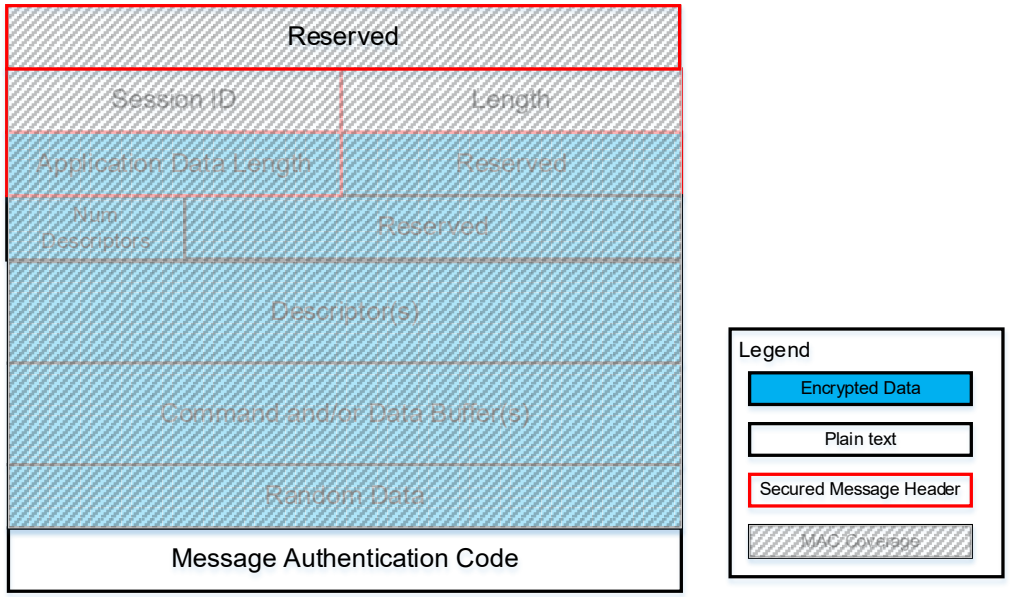
- SPDM defines several transcript hashes, including for CHALLENGE_AUTH, MEASUREMENTS, and KEY_EXCHANGE
  - The transcript hash is used to prevent MITM attacks and is fundamental to SPDM's security
  - Both the Requester and Responder build their own transcript hash
    - Signing only works if the two transcript hashes match
- The transcript is only over the SPDM command contents
  - Does not cover IF command, SPDM Storage Discovery or Pending Info
  - Does not include padding, if present
- Enables passthrough
  - The transcript can still be calculated, even if a protocol translation occurs, because SPDM commands are still used
  - Enables a passthrough command model
    - Controller just needs to support its protocol set and the caller can do the rest
  - Allows designing the storage controller out of the TCB

SPDM Secured Messages

- SPDM Secured Messages (DSP0277) are optional
  - The secured session is set up using DSP0274 defined commands (SPDMOperation = 5)
  - SPDM Secured Messages are sent using this binding (SPDMOperation = 6)
- Types of commands that can be conveyed
  - SCSI
  - ATA
  - NVMe
  - SPDM (DSP0274)
- Not optimized for efficiency
  - Not meant for disk IO

SPDM Secured Messages Format

- Transmits in the data buffer of an IF-SEND or IF-RECV
- Carries messages from DSP0277
- 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



Status Reporting

- There is a status hierarchy
  - The binding spec details how the status hierarchy works
- From highest priority to lowest priority (lower priority statuses are not reported)

1. Transport status – the status of the bus transfer, i.e. SAS transfer error
2. SPDM Storage Protocol Status – uses the bus status field to report status of the IF command, i.e. invalid field
3. SPDM protocol status – the SPDM ERROR response
4. Secured message encapsulated status – the status for the encapsulated message in a secured message (see Secured Command Status slide)

SPDM PQC Support

Background

- In August 2023, NIST published drafts of PQC contest winning algorithms.
  - (FIPS 203) "Module-Lattice-Based Key-Encapsulation Mechanism Standard" (ML-KEM); replacing Diffie-Hellman (aka Kyber).
  - (FIPS 204) "Module-Lattice-Based Digital Signature Standard" (ML-DSA); replacing RSA and ECDSA (aka Dilithium).
  - (FIPS 205) "Stateless Hash-Based Digital Signature Standard"; replacing RSA and ECDSA (aka SPHINCS+)
  - Final specifications[1] published August 2024
  - Another PQC signature winner but no public draft yet: Falcon
  - NIST is still looking for more digital signature schemes, preferably not based on Module-Lattice.
- CNSA specs are expected that will make these required.
  - While not every country will follow CNSA requirements, many will

Proposed Plan - Core Messages Adopting PQC

- Immediate Need is Signatures
  - Adopt PQC for the scenario where the public key is pre-provisioned to peer.
  - Adopt PQC signatures after X.509 cert supports PQC (RFC expected by end of 2024).
  - Adopt PQC key encapsulation after SP 800-227, which indicates requirements for ML-KEM in protocol implementations, is published.
  - This is planned to be SPDM 1.4, released on or after 4Q24.
    - Any changes/features that happen to be ready and merged will also be included.
- Consider adopting PQ/T (hybrid) signature and key encapsulation schemes.
  - Once the industry has general agreement on how.
  - This addition may be captured in a later SPDM release.
- libspdm support
  - lbspdm is likely to work on 1.3 through 4Q24.
  - Plan of record will be to add PQC subsequent to being feature complete for 1.3.
    - Given that SPDM doesn't implement any algorithms and instead references libraries, this is not expected to be burdensome.

Impact of PQC on SPDM

- SPDM message length is 2 bytes (64K)
  - In order to support SPINCS+, length needs to be extended.
  - Kyber & Dilithium can fit within 64K packet length
- Extend data structures
  - Handle larger lengths without reformatting the packet
    - In most cases, a bit will indicate whether to use the old length field or a new, larger field in the packet.
  - New Algorithm Structures
    - Separation of the traditional algorithms from PQC in the algorithm negotiation structures.
    - Leave room for the newer algorithms expected.
  - Hybrid solution will depend on the industry
- May need additional commands to support PQC algorithms
  - Examination of impact of algorithms for any semantic changes

PQC effect on SPDM over MCTP

- Dilithium over MCTP on I2C is problematic
  - I2C has 64 byte messages, leaving 51 bytes per SPDM payload
  - I2C is commonly at 100kpbs which is 195 messages/second
  - SL-DSA-SHA2-256f (SPHINCS+) signatures are expected to be as big as **49856 bytes**
    - ML-DSA-87 is 4627 bytes
  - That is roughly 4.7 seconds per SL-DSA-SHA2-256f signature which SPDM would use for MEASUREMENTS
    - And .46 seconds for ML-DSA-87
  - HPE is seeing 5 measurements for some NVMe drives, but as high as 11 for some drives (and worst case, 72 measurements for extreme devices).
  - That's 23.6 seconds additional boot time per drive for servers that can transmit MCTP messages (and there are some) for 5 signatures.
    - But still 2.3 seconds in the 5 signatures case for ML-DSA-87 and as high as 33.5 seconds.
- Realistic path for the short term
  - For I2C, perform traditional SPDM on I2C for boot time I2C and use MCTP over PCIe VDM for SPDM PQC where available.
  - Assuming devices & BMC have the processing power & buffer space for PQC (not a safe assumption)
- What the industry needs for PQC on the control plane (out of band)
  - Move to I3C/USB for device management.
    - Would be nice if OCP devices would align on just one (preferably USB)
  - Implement algorithms in silicon

Call to Action

- Understand what SPDM means to your ecosystem.
- Get ready for SPDM for storage
  - Review the Work in Progress release when available and provide feedback
    - https://www.dmtf.org/standards/feedback
- Get ready for PQC!
  - I2C is insufficient for PQC
  - Move to I3C/USB for device management will alleviate this.
    - Would be nice if OCP devices would align on just one (preferably USB)
  - Implement algorithms in silicon

Please take a moment to rate this session.

Your feedback is important to us.

