STORAGE DEVELOPER CONFERENCE

SDC 23

*BY Developers FOR Developers*

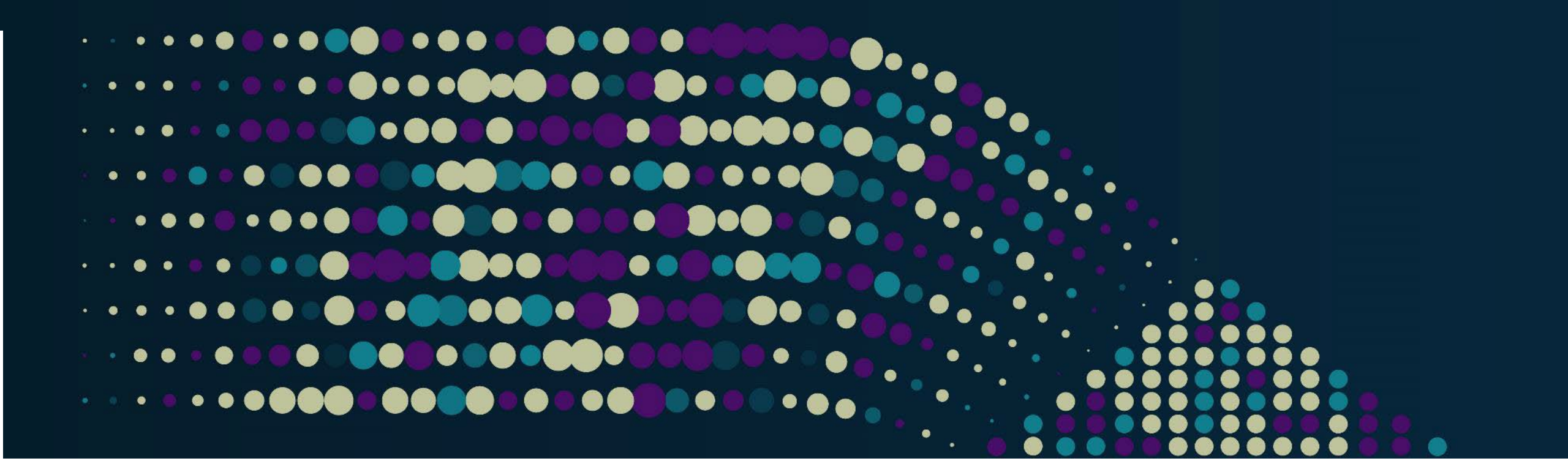
# NVMe over Fabrics Security Update

Claudio DeSanti
Distinguished Engineer
Dell Technologies CTIO Group

# Agenda

- SAN Security Framework
- NVMe/TCP with TLS
- TP 8018: Updates to NVMe/TCP with TLS
  - PSK Scope Confusion
  - TLS Concatenation use of Opportunistic TLS
- TP 8025: Usage Configuration of NVMe/TCP Security

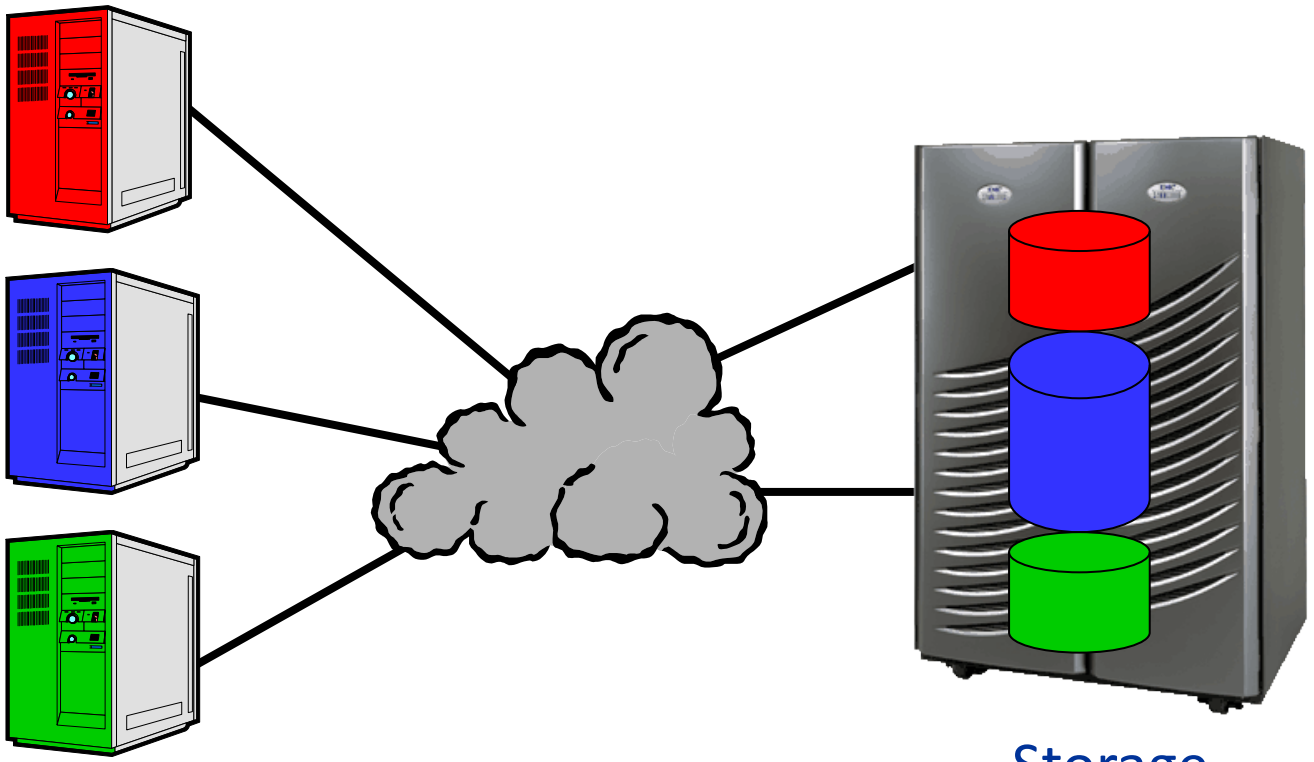
 



# SAN Security Framework

# Storage Area Network (SAN) Example

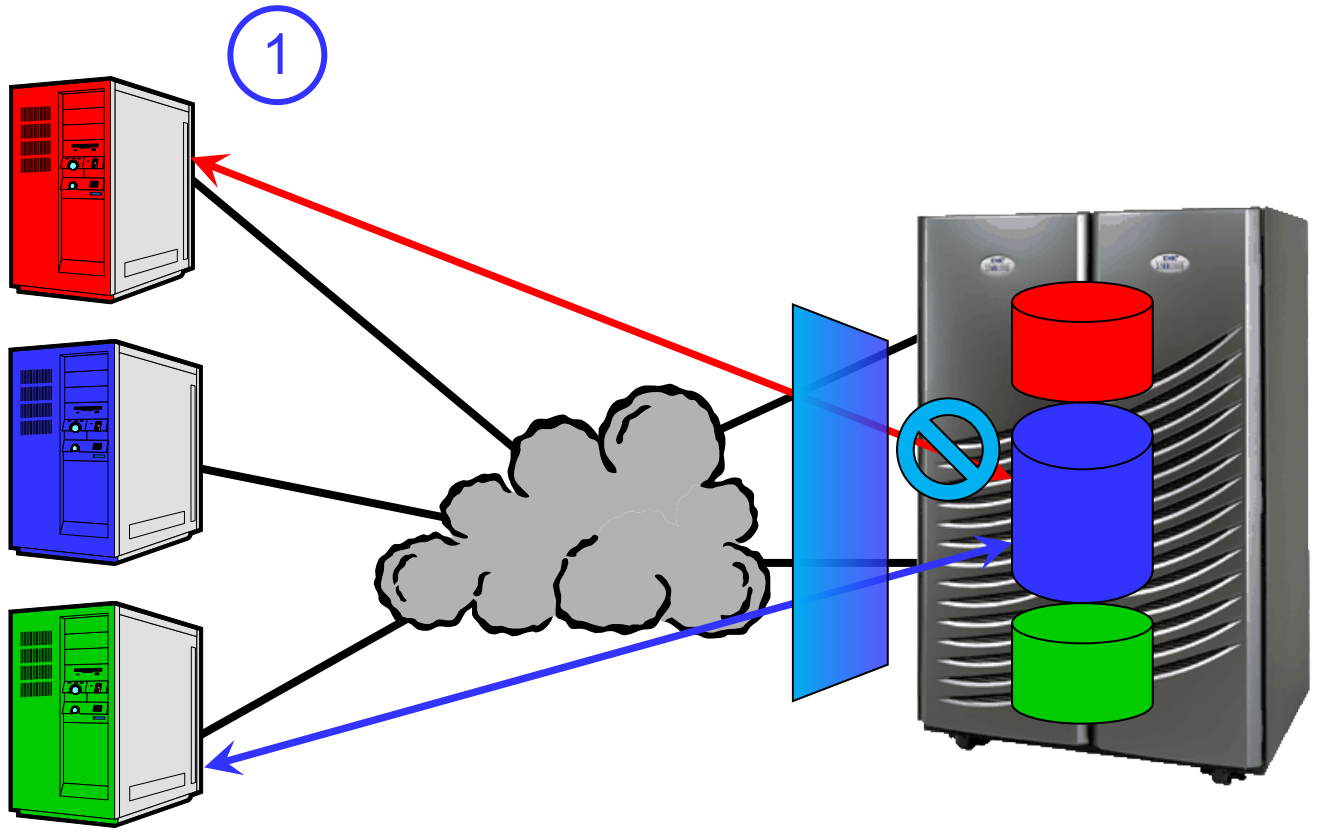


# Security Threat 1: Access Control

1) Uncontrolled Storage Access

- Countermeasure: Storage Access Control
  - NVMe namespaces mapping
  - NVMe-oF Zoning
- Does not prevent impersonation

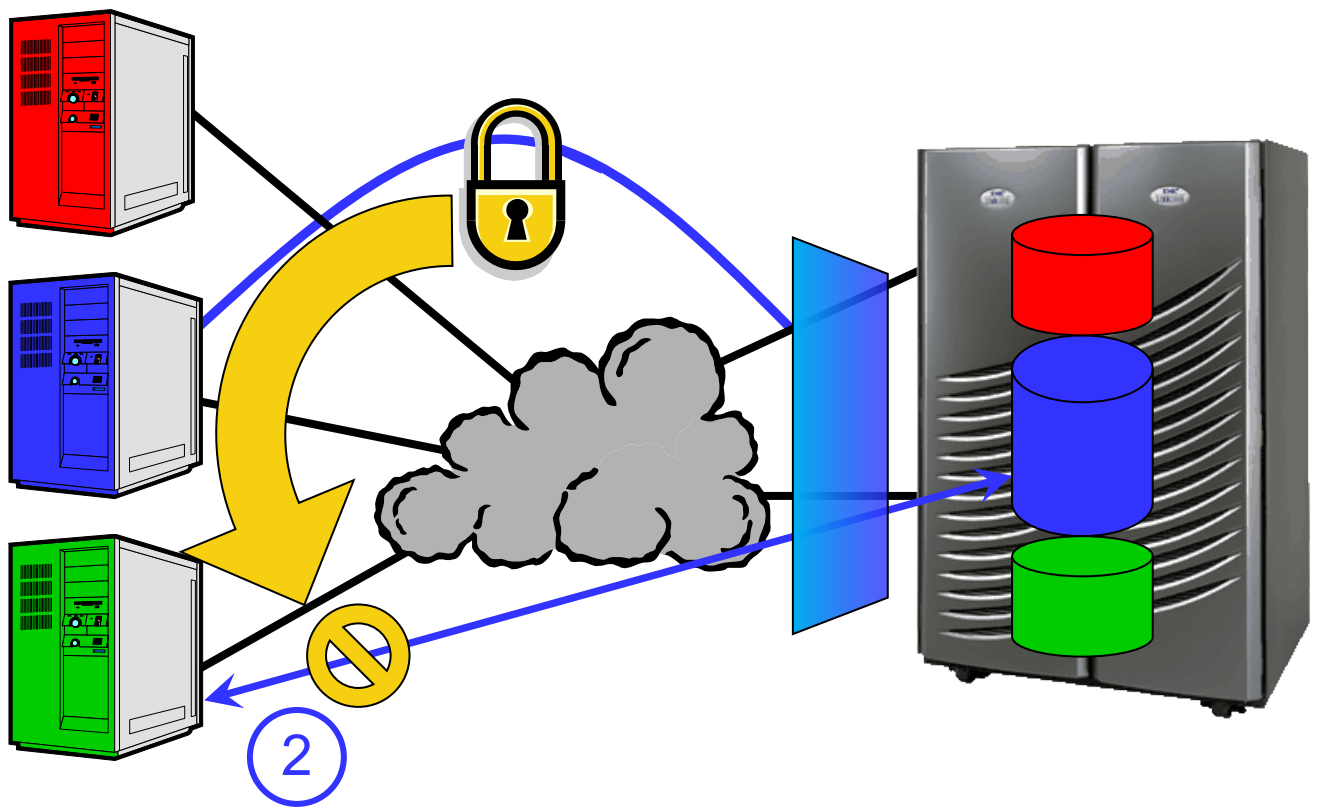


# Security Threat 2: Impersonation

2) Impersonation (Spoofing)

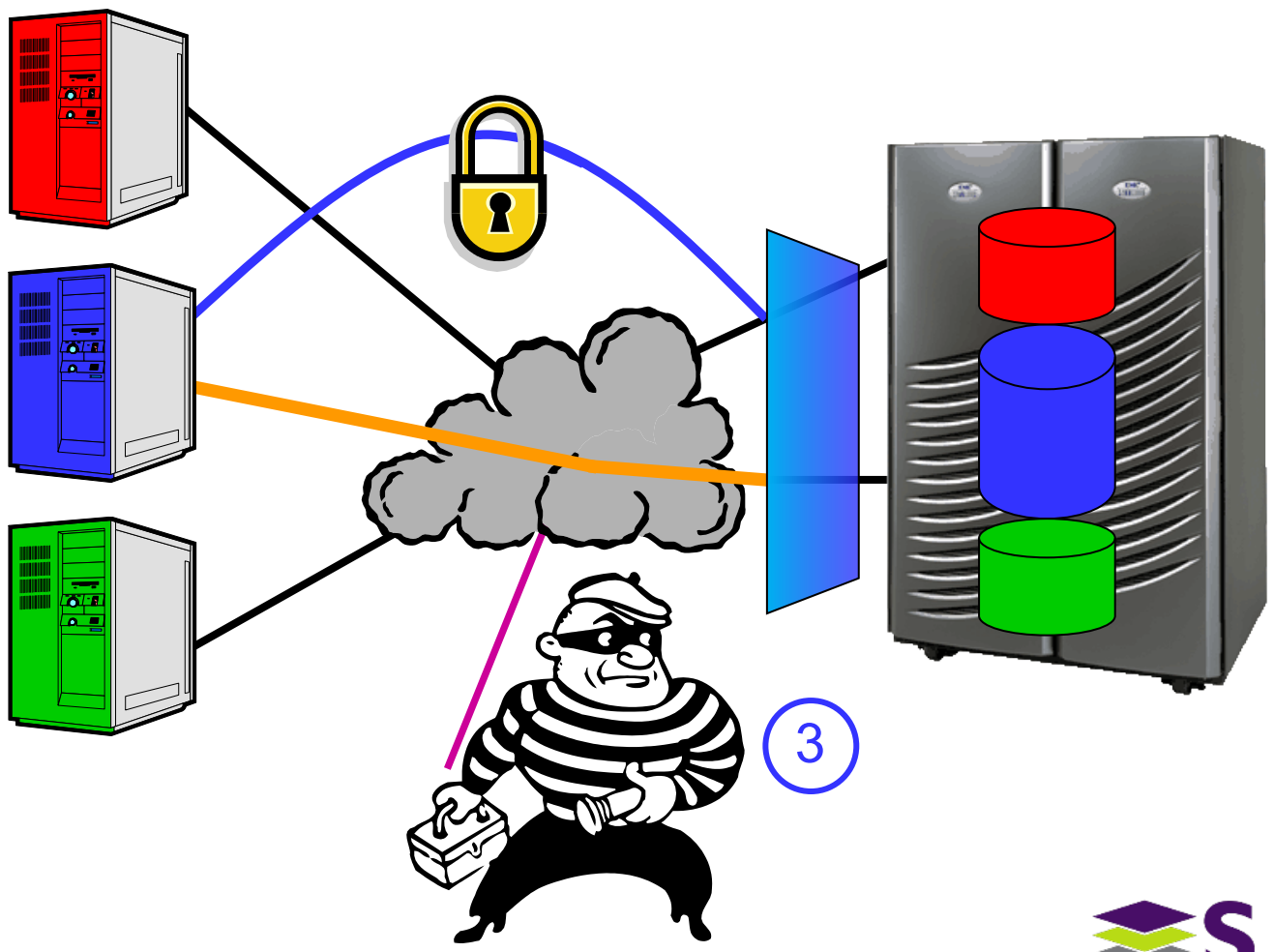
- Countermeasure: Authentication
  - Proof of identity



# Security Threat 3: Communication Access

3) Communication Access
   - Eavesdrop
   - Inject/Modify
- Countermeasure: Secure Channel (data in flight)
   - Confidentiality
   - Cryptographic Integrity

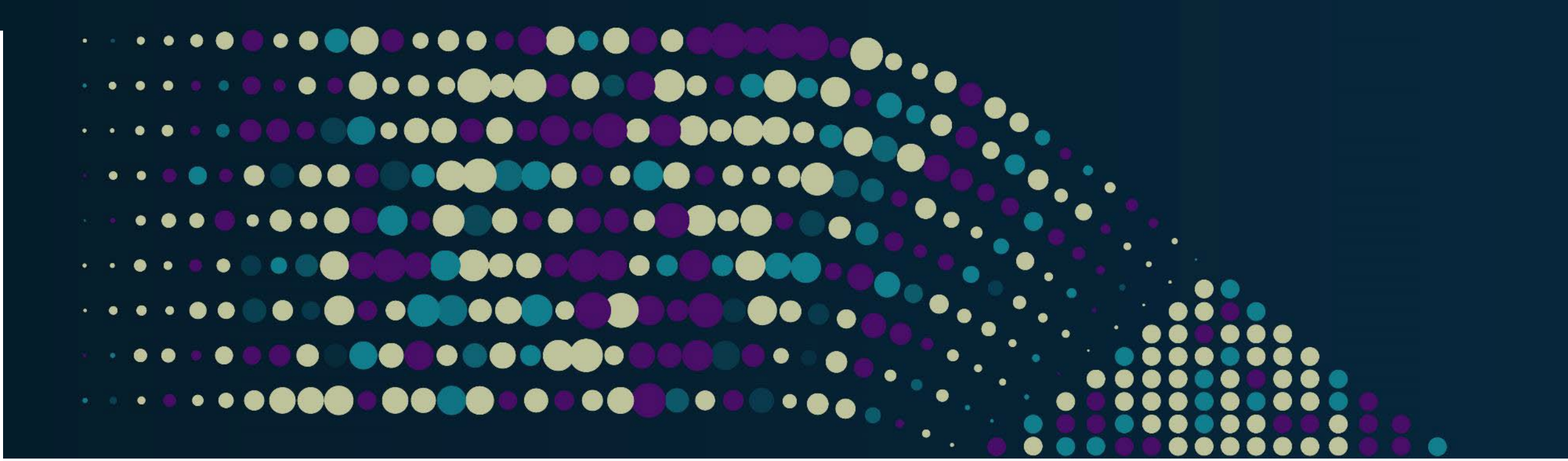


# SAN Security Mechanisms

| | iSCSI | Fibre Channel | NVMe over Fabrics/IP |
|---|---|---|---|
| **Storage Endpoint Authentication** | **CHAP** (strong secret)<br>**SRP** (weak secret, e.g., password)<br>[not used in practice] | **DH-CHAP** (strong secret)<br>**FCPAP** (weak secret, e.g., password)<br>**FCAP** (certificates)<br>**FC-EAP** (strong secret) | **DH-HMAC-CHAP** (strong secret)<br>- Defined in TP 8006<br>- Now in Base Spec. rev 2.0 |
| **Centralized Authentication Verification** | **RADIUS** (CHAP-only, obsolete) | **RADIUS** (DH-CHAP-only, obsolete) | Authentication Verification Entity (**AVE**)<br>- Defined in TP 8019 |
| **Secure Channel** (authenticated encryption & cryptographic integrity) | **IPsec** (e.g., in security gateway) | **FCsec** (IPsec-like, usage limited) | **TLS** (pre-shared key) – for TCP only<br>- Defined in TP 8011<br>- Now in NVMe/TCP Transport Spec. |



# NVMe/TCP with TLS

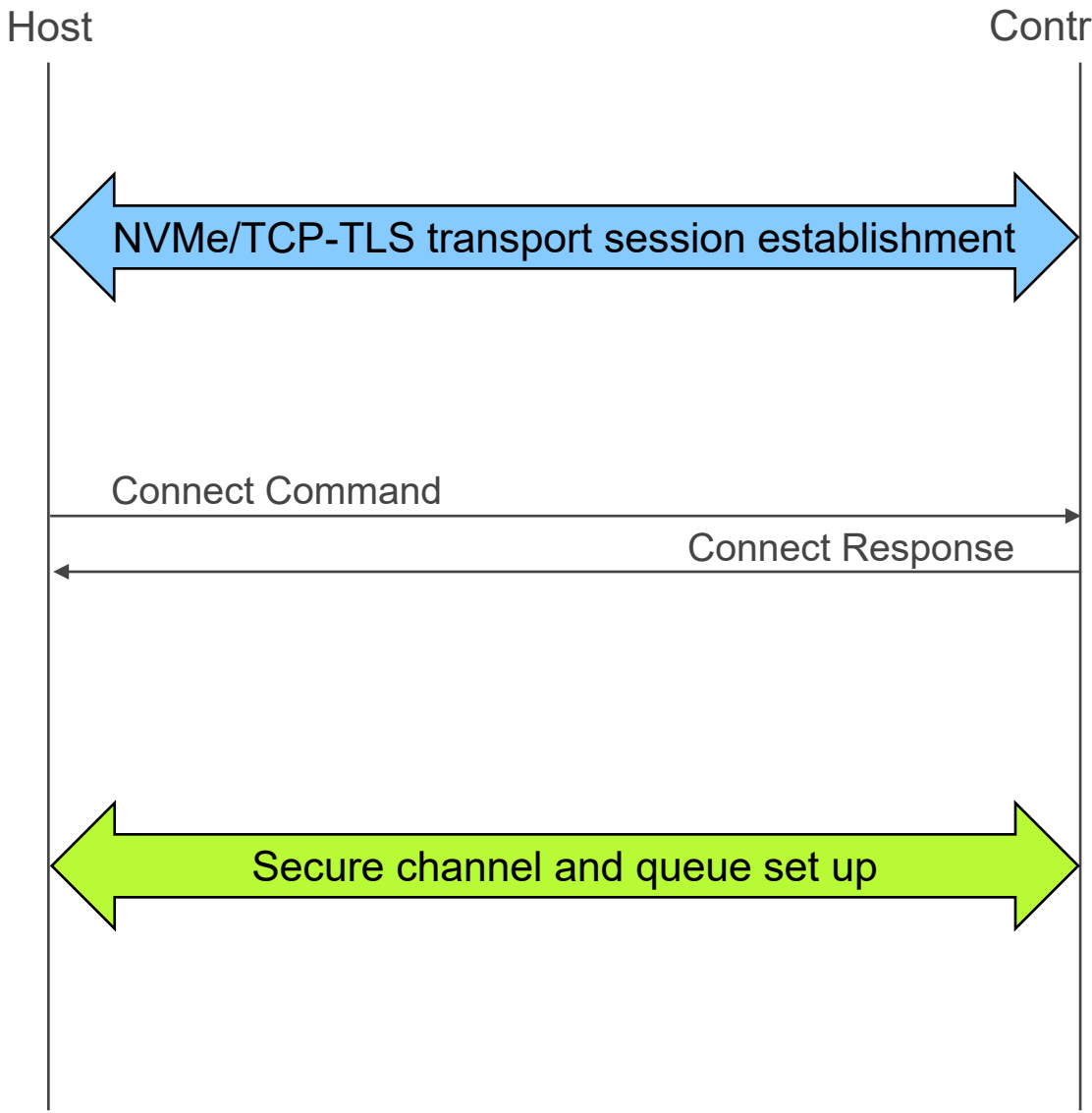
 

# NVMe/TCP Secure Channel: TLS

- TLS (Transport Layer Security): widely used TCP secure channel protocol
  - Secure channel = authentication, confidentiality, cryptographic integrity (primary properties)
  - Typical (web) usage: server uses certificate with TLS, client authenticates after TLS setup (e.g., TLS-protected HTTP)
- TLS 1.3 for NVMe/TCP: part of NVMe TCP transport specification
  - TLS not specified for other NVMe-oF IP-based protocol (i.e., RDMA, e.g., RoCEv2)
  - TLS not implemented in NIC RDMA hardware data paths, hence not usable in practice
- Older TLS versions:
  - TLS 1.2: Strongly discouraged in favor of modernized, more secure TLS 1.3
  - Older versions of TLS (1.0 & 1.1) & all versions of SSL: insecure, hence prohibited

# NVMe/TCP-TLS




1. An NVMe/TCP-TLS transport session is established
2. The Connect exchange is performed to set up NVMe Queue and associate host to controller
3. Secure channel and Queue are set up, ready for subsequent operations

# TLS Credentials (1): Not X.509 Certificates (for now)

- NVMe X.509 certificate identities: use NVMe-native identities
  - NVMe supports non-IP transports (e.g., Fibre Channel, InfiniBand) that do not use web or IP identities
    - SPDM-based certificate authentication anticipated for all NVMe transports
  - NVMe certificates should use NVMe-native identities (i.e., NQNs, text strings similar to iSCSI IQNs)
    - Reminder: certificate binds identity to public key, binding is signed by private key of Certificate Authority (CA)
    - Mapping certificate identity to actual identity weakens security because that mapping is not signed
  - Requires an NVMe-specific X.509 certificate format to use NVMe NQNs as certificate identities

- Certificate lifecycle management (e.g., issuance, revocation, replacement)
  - Q: Who operates the CAs (Certificate Authorities) that issue and revoke certificates?
  - A: Vendors, including OEMs (e.g., Dell, HPE) and device vendors (e.g., Samsung, Kioxia, Seagate)
  - Problem: That's not how existing Internet/web CAs are operated
  - PCIe is blazing a trail in this area, NVMe chose to wait, watch, and learn

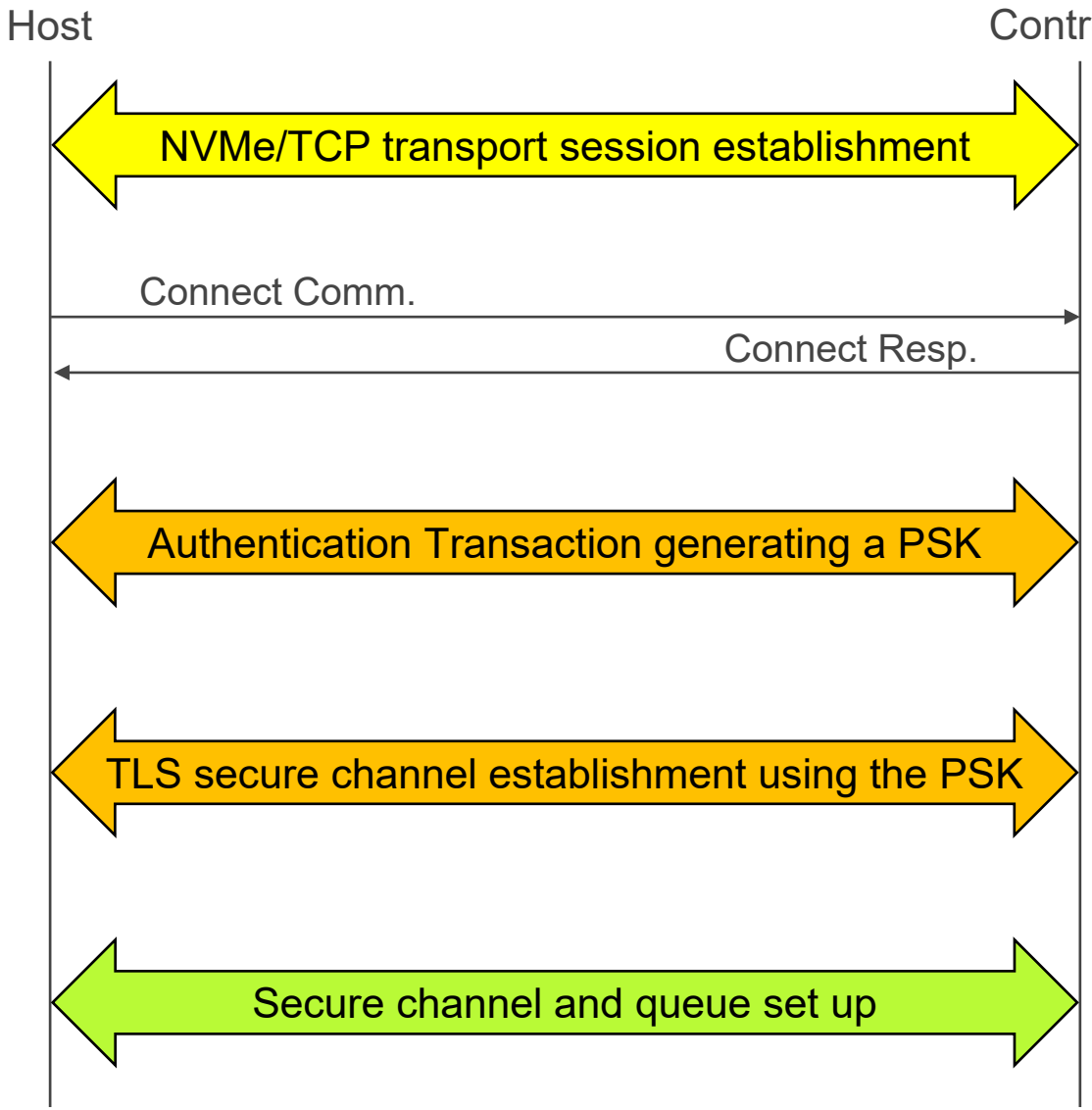
# TLS Credentials (2): Pre-shared Keys

- TLS secure channel for NVMe/TCP is based on pre-shared keys (PSKs)
  - In order to communicate over TLS, two NVMe entities need to be configured with the same PSK
  - Each pair of entities require its own PSK (limits “blast radius” of PSK compromise)
  - $O(n^2)$ problem: a fabric with N hosts and M subsystems requires N x M PSKs

- NVMe-oF authentication protocol (DH-HMAC-CHAP) to the rescue
  - Upon successful completion of an authentication exchange, the two involved NVMe entities generate an ephemeral shared session key (e.g., a ‘PSK’ computed on the fly)
  - The TLS negotiation can then be performed using a PSK derived from that shared key
  - Reduces TLS PSK provisioning to per-entity DH-HMAC-CHAP secret provisioning
  - O(n) problem: a fabric with N hosts and M subsystem requires N+M secrets
    - When AVE is deployed

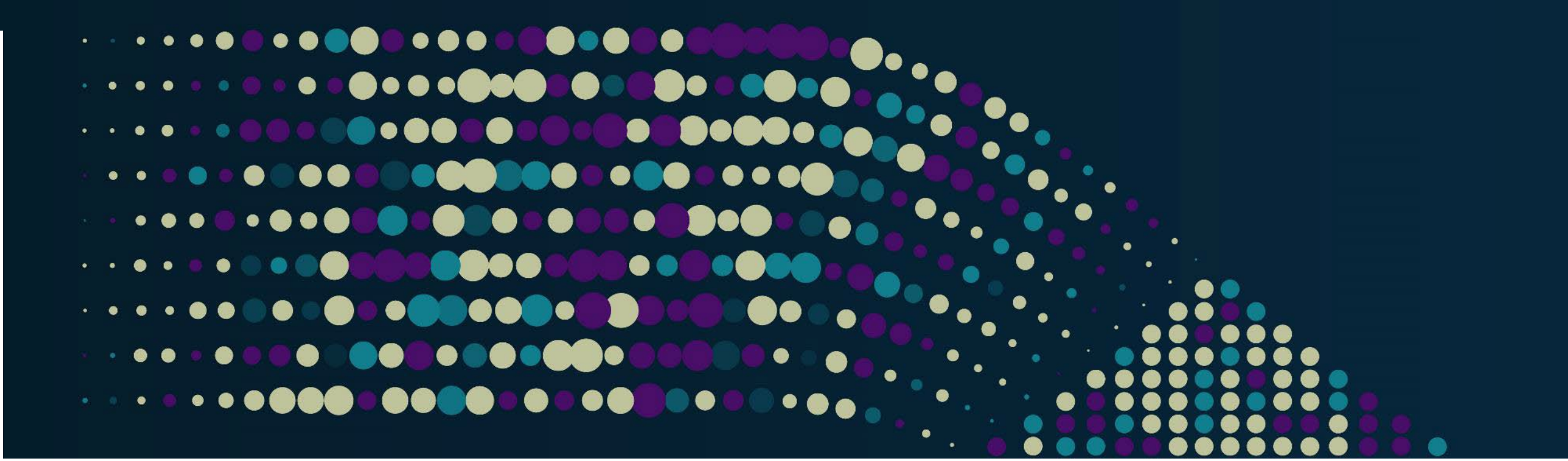
 

# (Old) TLS Concatenation




1. An NVMe/TCP transport session is established
2. The Connect exchange is performed to set up NVMe queue and associate host to controller
3. The host performs an authentication transaction with the controller, transaction that generates a pre-shared key PSK between host and controller
4. The generated PSK is used to perform a TLS negotiation and to establish a secure channel
5. Secure channel and queue are set up, ready for subsequent operations



# TP 8018: NVMe/TCP with TLS Updates

## PSK Scope Confusion

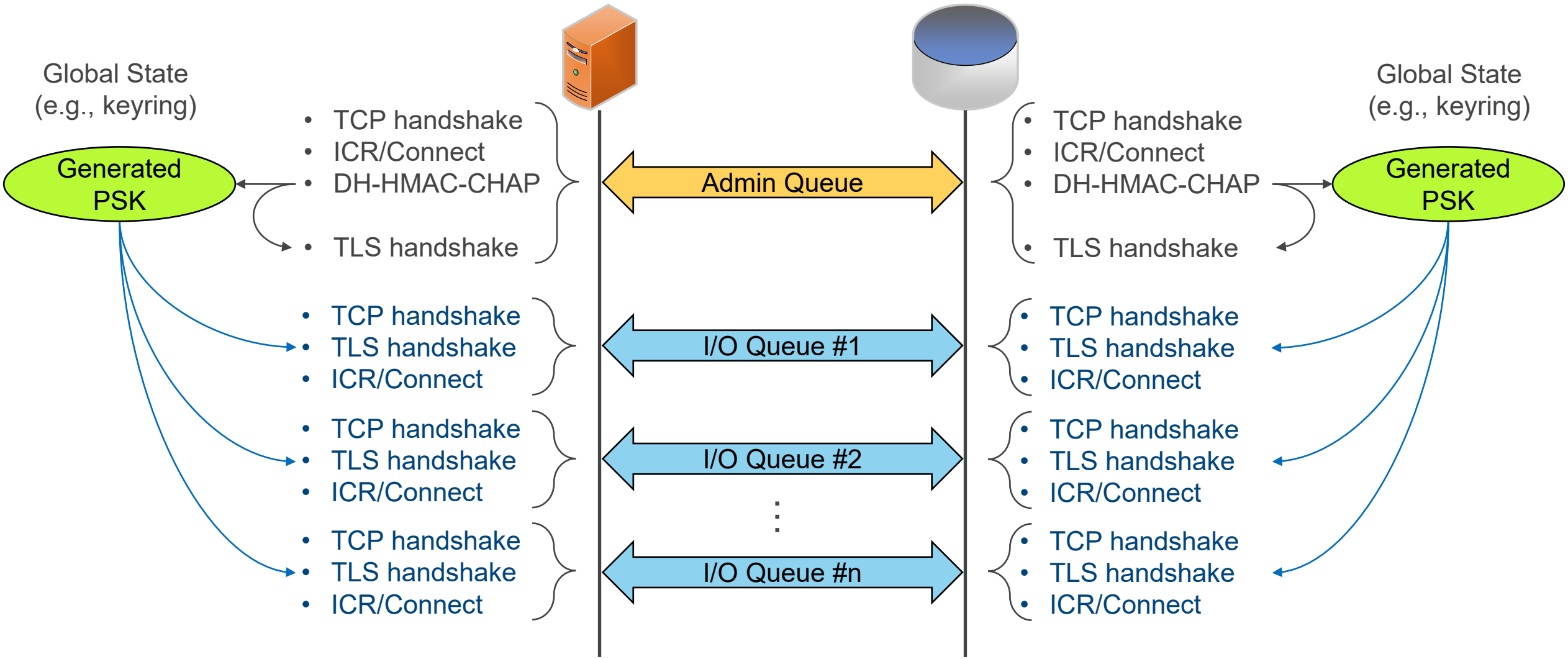
 

# TLS PSK Scope

- NVMe/TCP defined two methods for obtaining a PSK…
  - Retained PSK: from an administratively provisioned configured PSK
  - Generated PSK: from a DH-HMAC-CHAP authentication transaction
- …And two associated PSK identities
  - Because TLS 1.3 requires each PSK to be associated with one and only one PSK Identity
  - “NVMe0R<hash> <$NQN_h$> <$NQN_c$>”
  - “NVMe0G<hash> <$NQN_h$> <$NQN_c$>”
- The scope of a retained PSK was defined per entity pair
  - All TLS sessions between a host and an NVM subsystem use the (single) retained PSK
- The scope of a generated PSK was not well defined
  - Per Admin Queue or I/O Queue?
  - Per NVMe/TCP controller?
  - Per entity pair?
- This caused confusion (next slides)

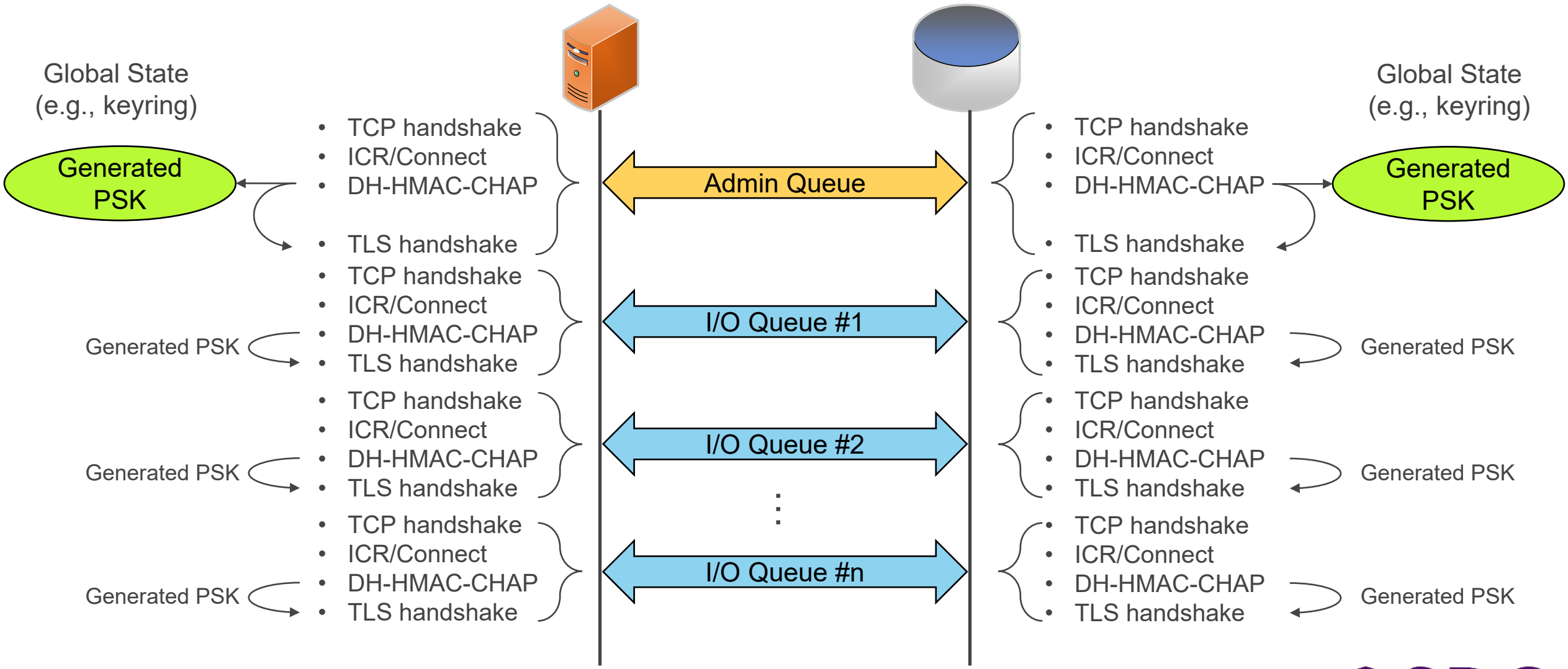
 

# Use Case #1: Per Controller Scope



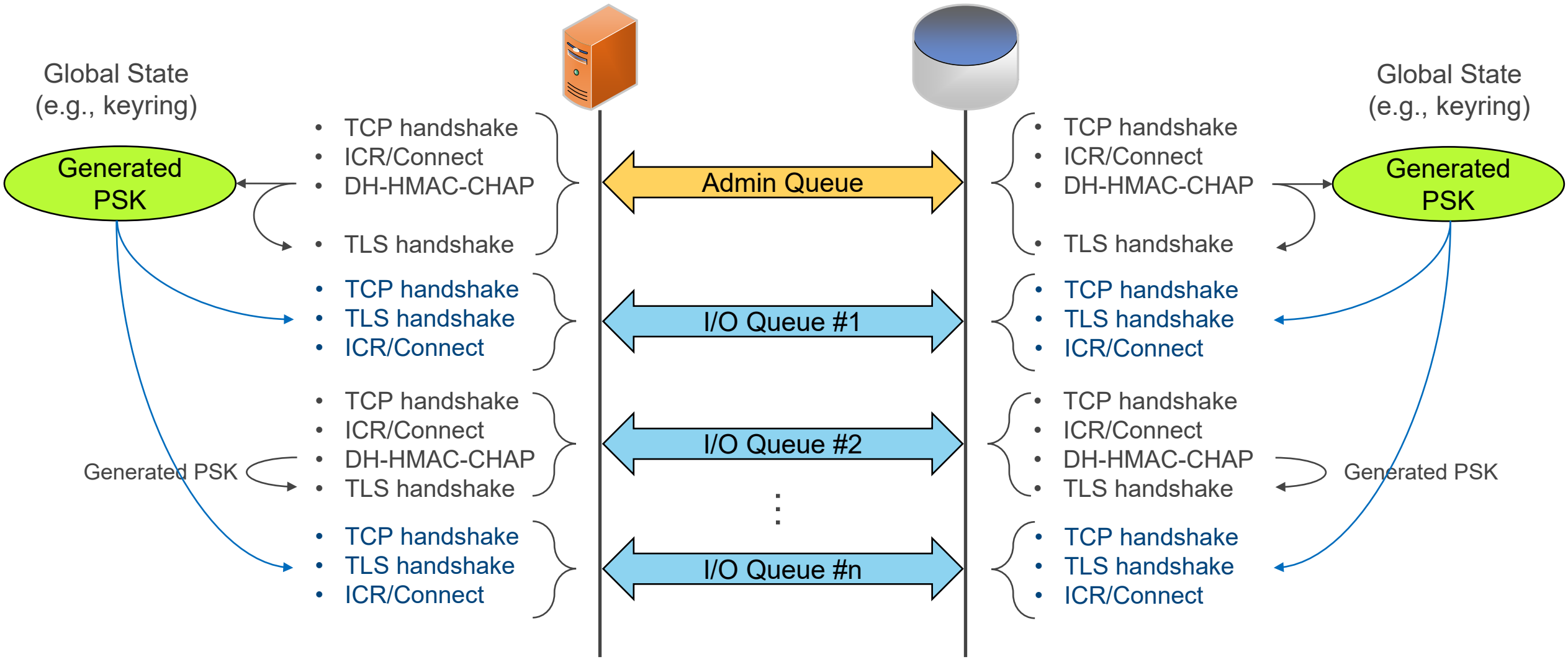

 

# Use Case #2: Per Queue Scope



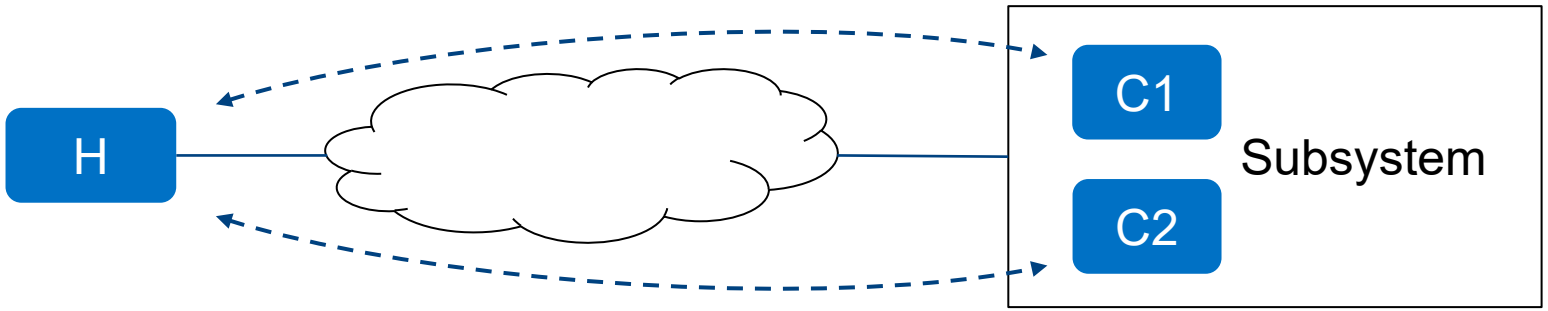
# Use Case #3: Mixed Scope



# Scope Confusion and Solution

- Having a single PSK Identity for all generated PSKs between two entities is problematic
  - Use case #2 can leverage the TCP connection for disambiguation
    - Can manage generated PSK as ephemeral-only
- Use case #1 is the one desired, where a PSK is reused across all connections to the same controller
  - Cannot manage generated PSK as ephemeral-only because reused
  - Cannot disambiguate associations to different controllers




- Solution Approach (TP 8018): Two Parts
  - A. New Per-PSK unique identities
    - Support unambiguously multiple scopes of a PSK
  - B. Restrict TLS PSK generation to Admin Queue (Use Case #1)
    - I/O Queues reuse PSK generated on Admin Queue

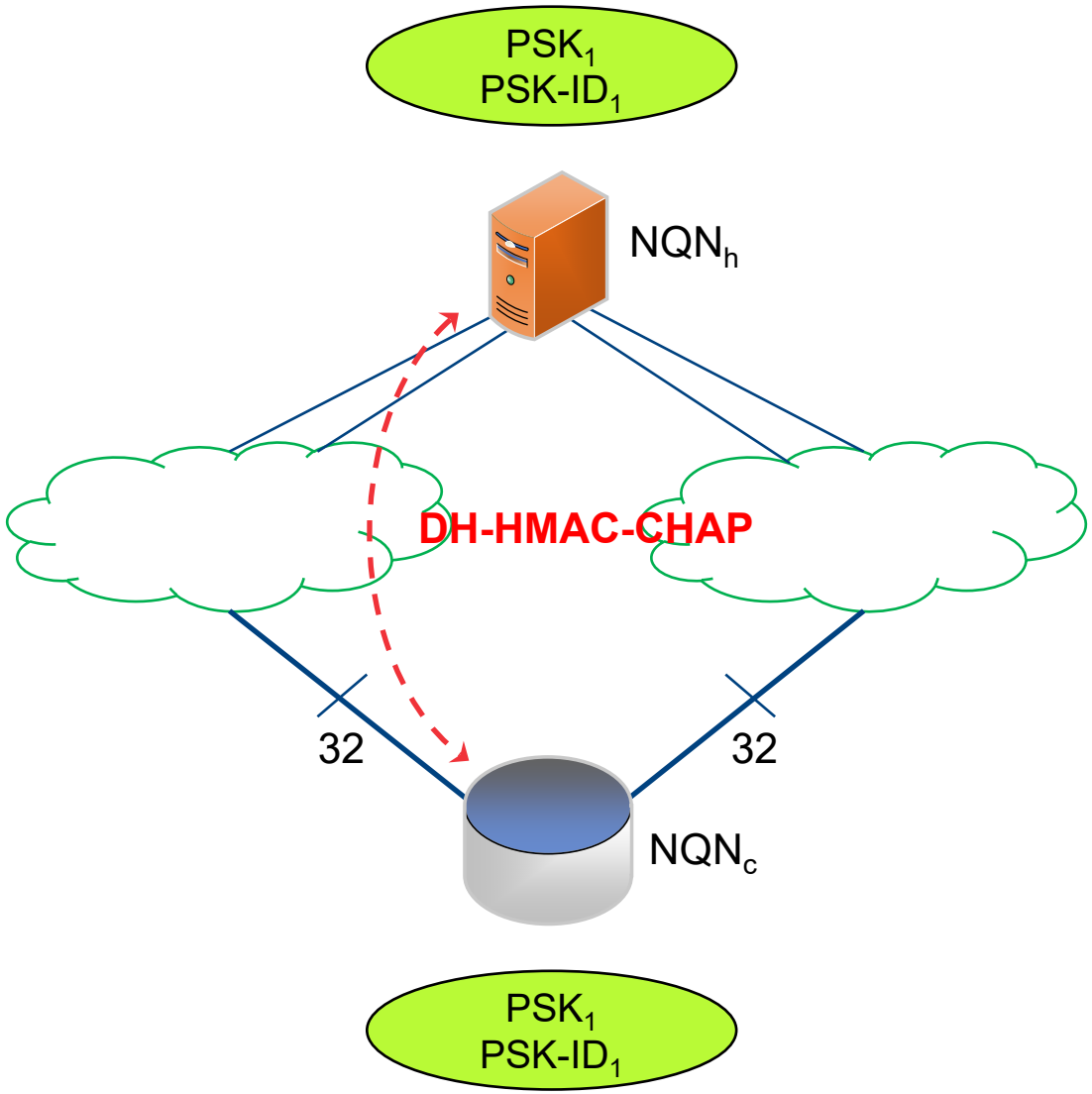
 

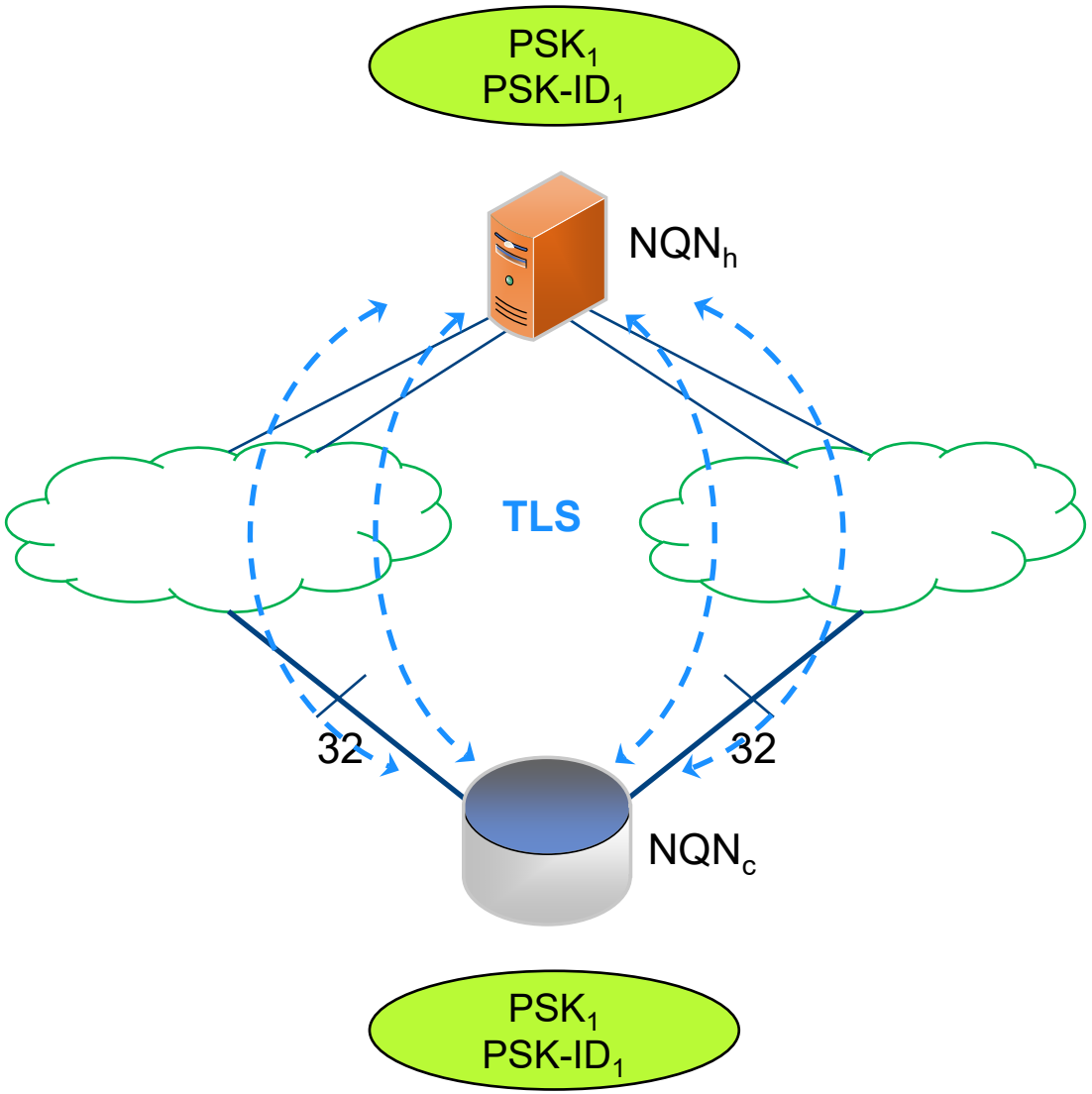
# Part A. New Per-PSK Identities: PSK 'Digest'

- Compute a PSK 'digest'
  - Use secure <hash> associated with PSK (in form of HMAC)
  - Digest computed from PSK, Host NQN, and NVM Subsystem NQN (as HMAC inputs)
    - And represented in ASCII via a Base64 conversion
- PSK Identity uniqueness: add the digest to the PSK Identity
  - "NVMe~~0~~1R<hash> $NQN_h$ $NQN_c$ <digest>"
  - "NVMe~~0~~1G<hash> $NQN_h$ $NQN_c$ <digest>"
  - Resulting PSK Identity uniquely identifies the actual PSK
    - Uniqueness: Hash (HMAC) output size is at least 256 bits, cryptographically binds PSK to its PSK Identity
    - A different PSK produces a different digest hence a different PSK Identity
- New functionality enabled by per-PSK Identities
  - Use different generated PSK on different controllers (PSK Identity uniquely identifies PSK)
  - Bonus: PSK rollover support – PSK identities distinguish old and new PSKs

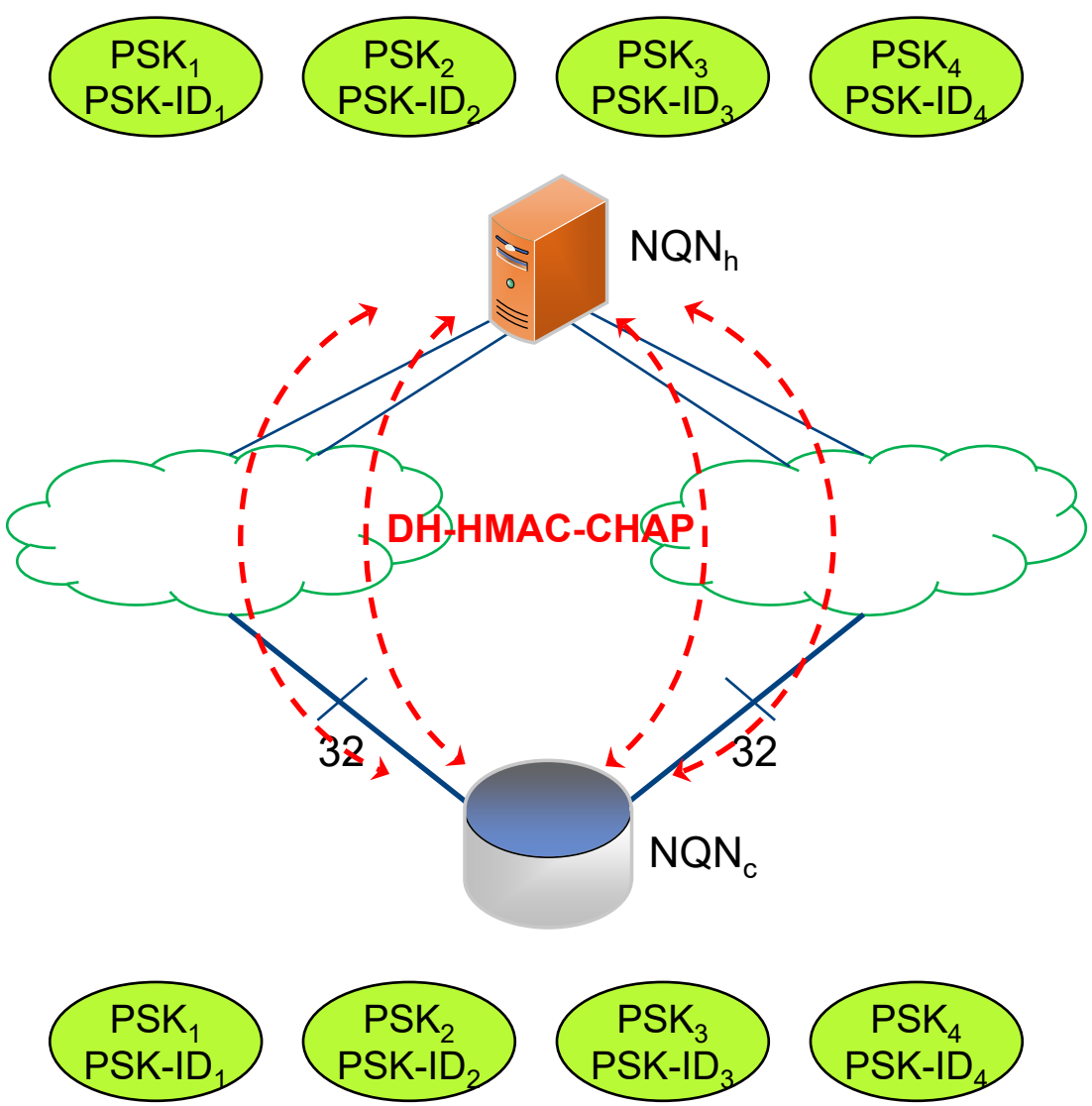
# Per Entity Pair Scope

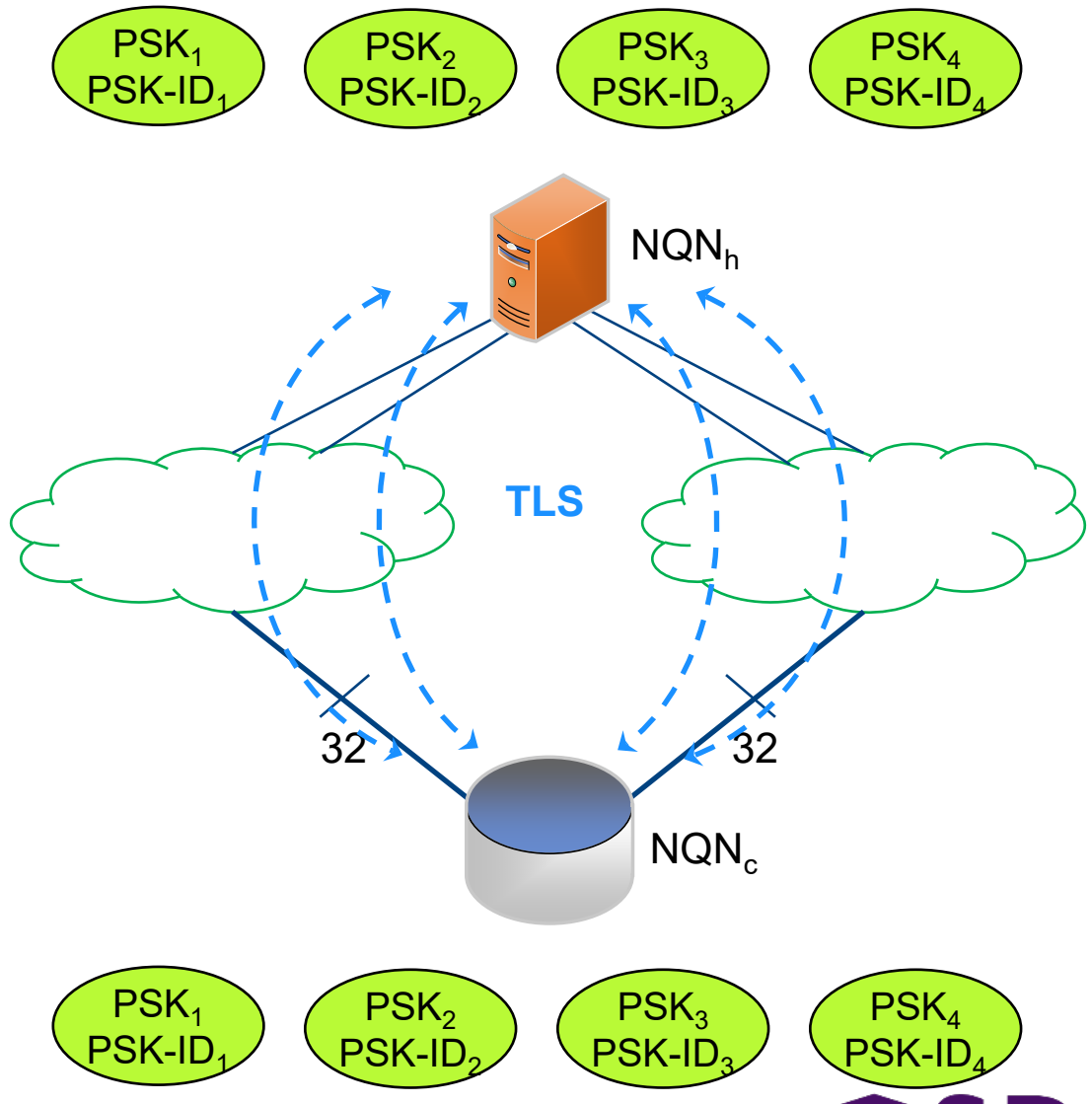


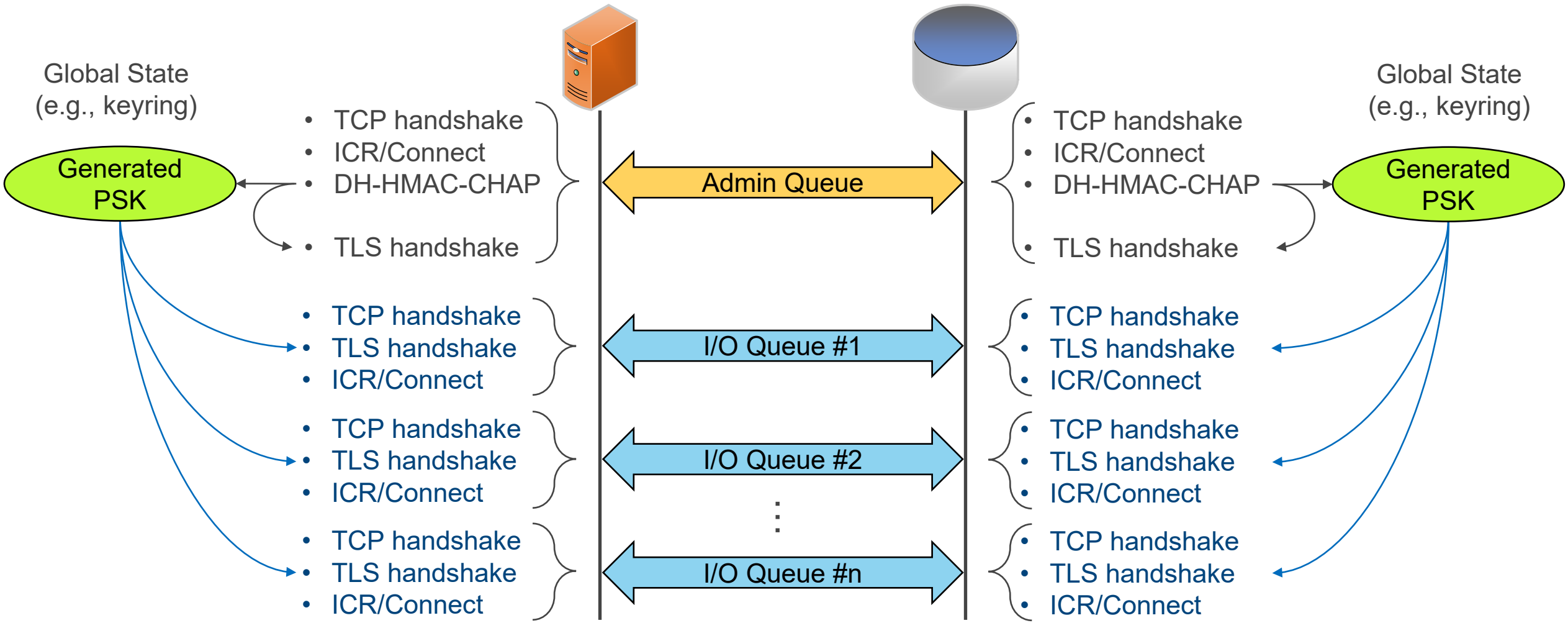
# Per Association Scope



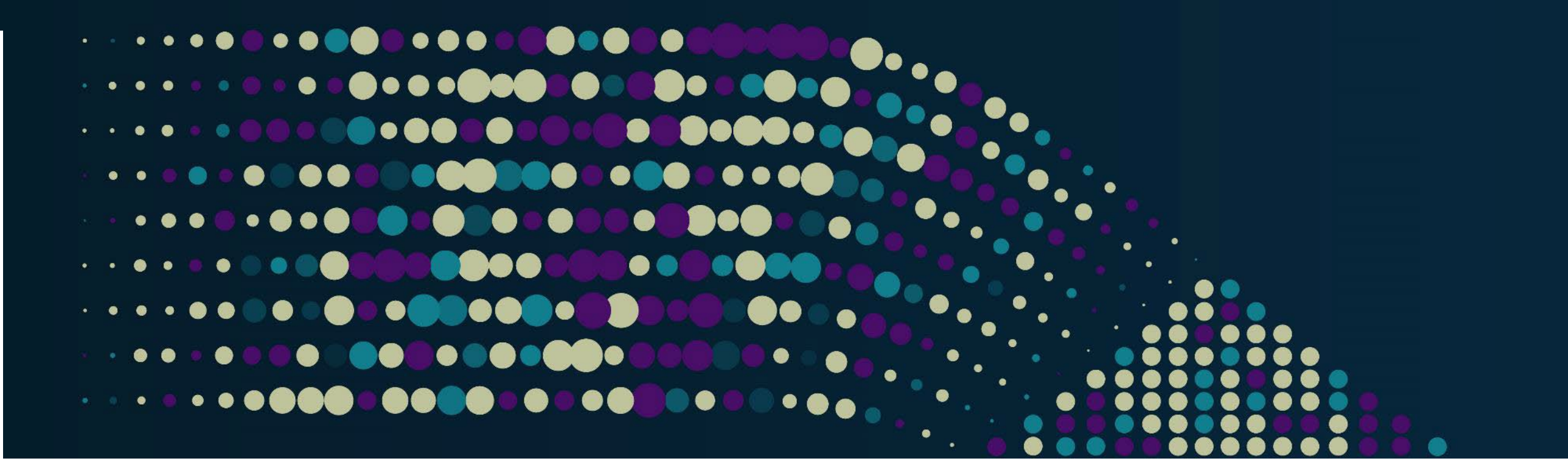

 

# Part B: Support Only Use Case #1




- New capability: Generated PSK can be shared among controllers



# TP 8018: NVMe/TCP with TLS Updates

## TLS Concatenation use of Opportunistic TLS

# A Funny Thing Happened…

…in an early NVMe/TCP TLS implementation

- Reminder: TLS concatenation switched to TLS in the middle of a TCP stream
  - DH-HMAC-CHAP authentication performed in the clear, generates a TLS PSK
  - Generated TLS PSK used to authenticate TLS handshake
  - TLS handshake enables TLS, TCP traffic continues under TLS encryption
    - This is called “Opportunistic TLS”
- Interesting theory… what about practice?
  - Implementer: What do I do with the NVMe commands that arrived during the TLS handshake?
  - Protocol designers: Ehmm???  That wasn’t supposed to happen!!
  - Implementer: But the code did that… now what am I supposed to do??
- 20/20 Hindsight: example of a known problem with opportunistic TLS
  - Unexpected non-TLS traffic bypasses TLS handshake in progress
  - Has been seen in SMTP implementations of TLS
    - Multi-layer protocol stack automatically directs incoming traffic to the right protocol layer

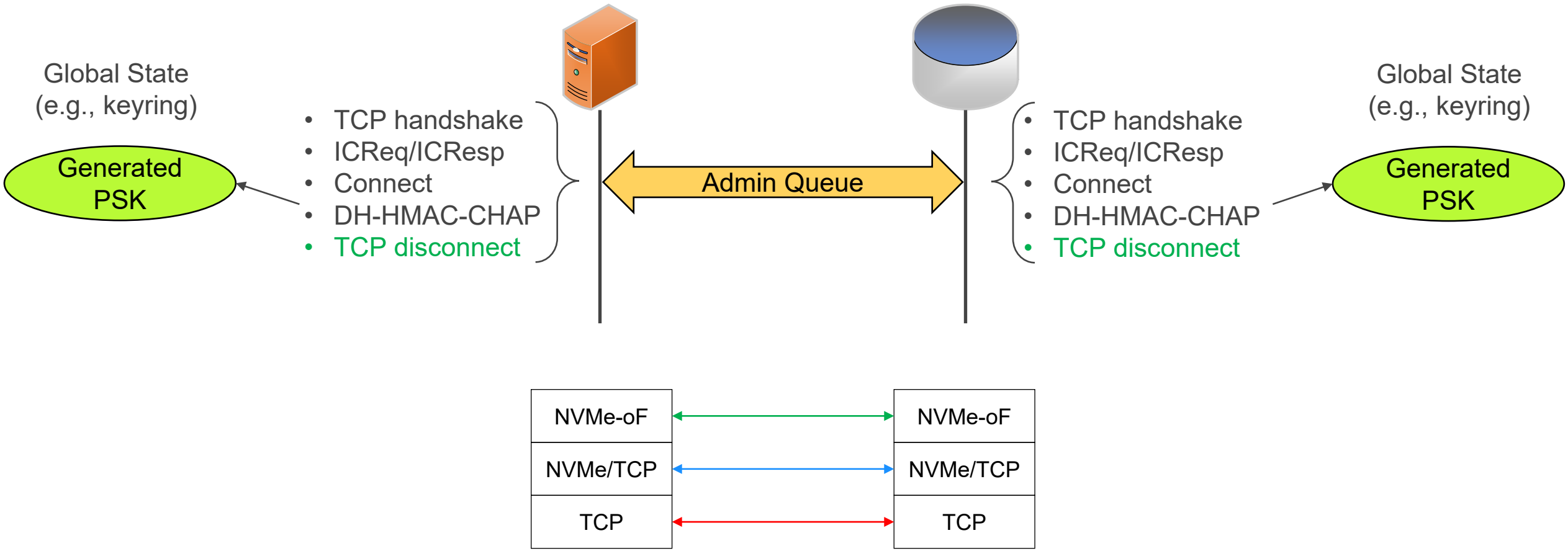
 

# Solution Approach

- Part A: Change to a second TCP connection
  - Disconnect: Remove first TCP connection after generating TLS PSK
  - Reconnect: Start second TCP connection, use TLS PSK to start TLS handshake
  - What happens to unexpected non-TLS traffic?
    - Stranded on first TCP connection (socket) because second TCP connection uses a different TCP source port
    - Bit-bucketed by teardown/cleanup of first TCP connection (socket) – robust assurance of discard
- Part B: Unexpected non-TLS traffic closes second connection
  - Simpler check because second TCP connection starts with TLS handshake
  - TLS alert sent (if appropriate) before closing TCP connection

# TLS Concatenation with Explicit Disconnect



# TLS Concatenation with Explicit Disconnect


Global State
(e.g., keyring)

Generated
PSK

- TCP handshake
- TLS handshake
- ICReq/ICResp
- Connect

Admin Queue

- TCP handshake
- TLS handshake
- ICReq/ICResp
- Connect

Global State
(e.g., keyring)

Generated
PSK

- TCP handshake
- TLS handshake
- ICR/Connect

I/O Queue #1

- TCP handshake
- TLS handshake
- ICR/Connect

- TCP handshake
- TLS handshake
- ICR/Connect

I/O Queue #2

- TCP handshake
- TLS handshake
- ICR/Connect

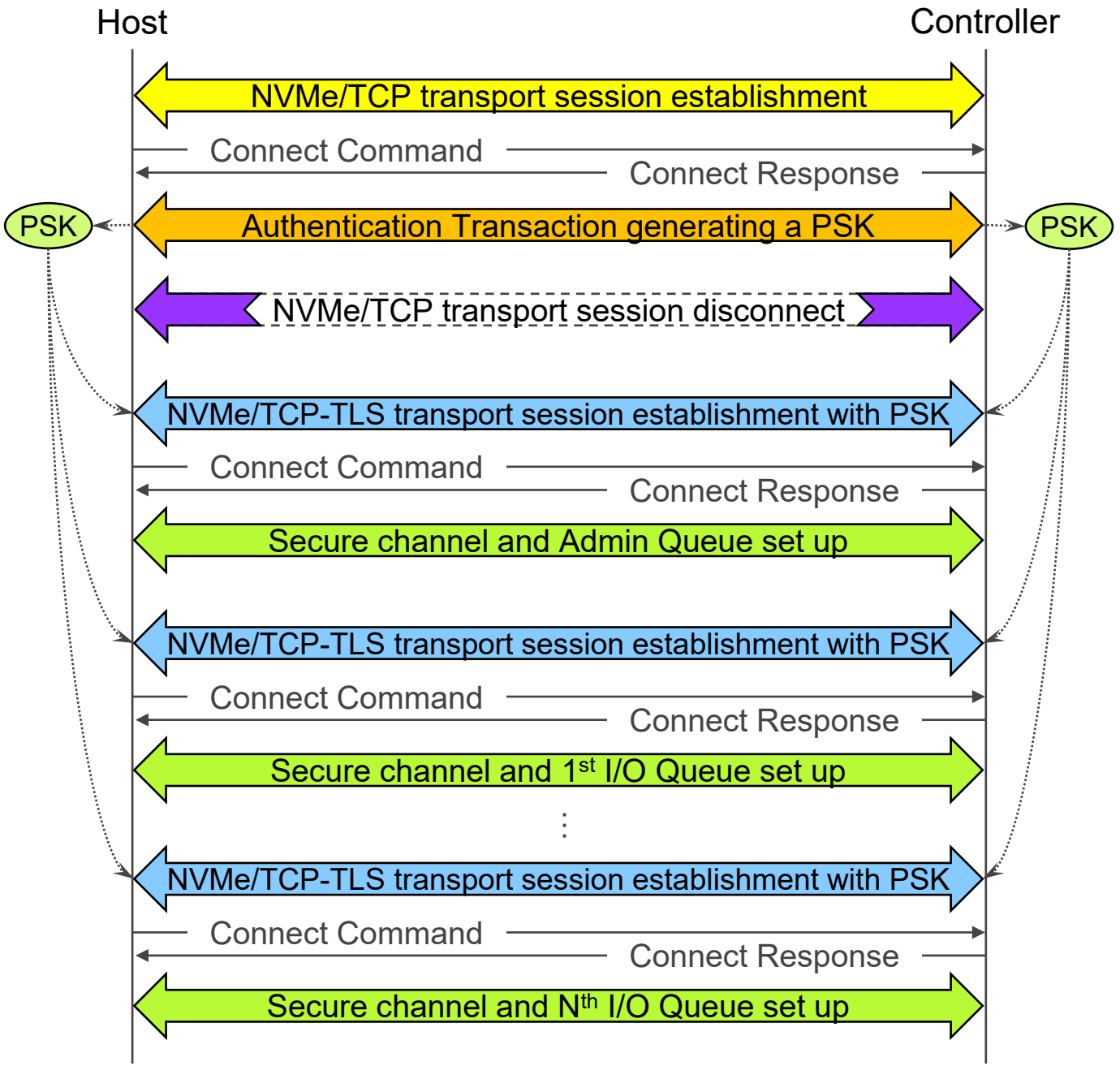
⋮

- TCP handshake
- TLS handshake
- ICR/Connect

I/O Queue #n

- TCP handshake
- TLS handshake
- ICR/Connect

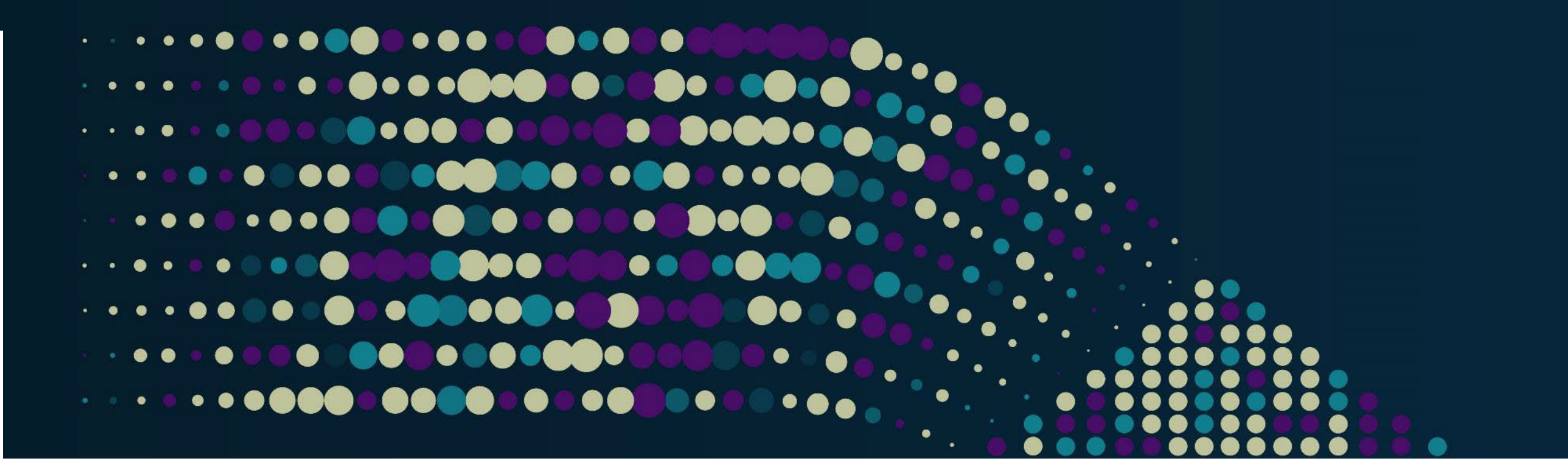



# Updated TLS Concatenation

1. An NVMe/TCP transport session is established
2. The Connect exchange is performed to set up an Admin Queue and associate host to controller
3. An authentication transaction generating a PSK between host and controller is performed
4. The NVMe/TCP transport session is disconnected
5. An NVMe/TCP-TLS transport session is established using the generated PSK
6. The Connect exchange is performed to set up an Admin Queue and associate host to controller
7. Secure channel and queue are set up
8. An NVMe/TCP-TLS transport session is established using the generated PSK
9. The Connect exchange is performed to set up the first I/O Queue of the established association
10. Secure channel and queue are set up

    ⋮
11. An NVMe/TCP-TLS transport session is established using the generated PSK
12. The Connect exchange is performed to set up the $N^{th}$ I/O Queue of the established association
13. Secure channel and queue are set up



# TP 8025: Usage Configuration of NVMe/TCP Security

## When to Use a Security Mechanism?

# Terminology

- Security mechanism **not provisioned**:
  - The parameters (e.g., DH-HMAC-CHAP secret or TLS Configured PSK) needed by that security mechanism have not been provisioned and hence the NVMe entity is not able to use that security mechanism
- Security mechanism **provisioned**:
  - The parameters (e.g., DH-HMAC-CHAP secret or TLS Configured PSK) needed by that security mechanism have been provisioned and hence the NVMe entity is able to use that security mechanism
- Once a security mechanism is provisioned, usage of that mechanism can be:
  - **Disabled**: do not use it
  - **Permitted**: negotiate with the other party
  - **Required**: do use it
- Defined in TP 8025

# Usage Behavior: General Logic

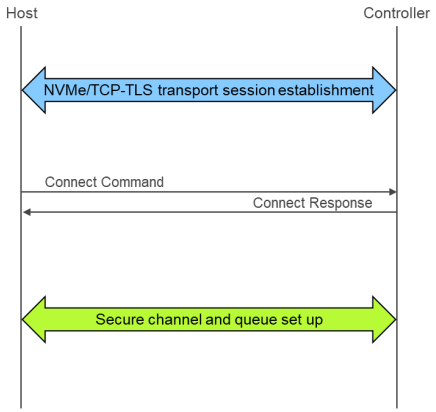
- If a security mechanism is disabled, then it is not used
  - No matter what the other entity configuration is
- If a security mechanism is required, then it is used
  - No matter what the other entity configuration is
- If a security mechanism is permitted, then:
  - The host decides about using the mechanism from the discovery log entry about the NVM subsystem
  - The NVM subsystem let the host decide

# Security Processing

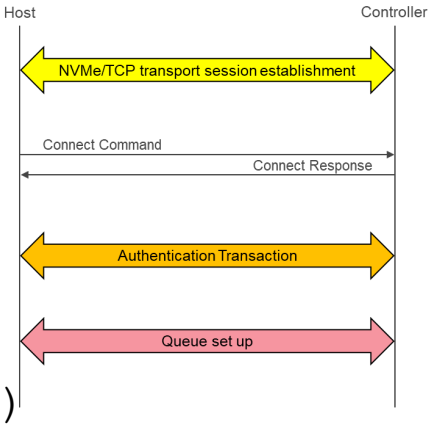
At TCP connection time:

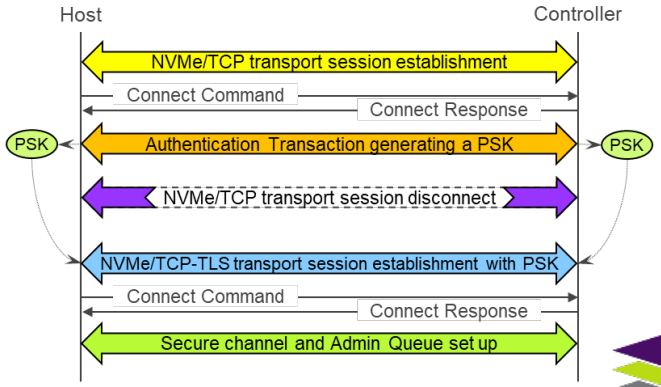
```
if (TLS provisioned)
   {follow TLS usage behavior}
else
   {continue}
```




After successful completion of the Connect command:

```
if (Authentication provisioned)
   {
   if (TLS not provisioned or connection already in TLS)
       {follow Authentication only usage behavior}
   else {follow Authentication with TLS concatenation usage behavior}
   }
else
   {continue}
```



# Example: TLS Usage Configuration

| Configuration | Description |
|---|---|
| TLS disabled | Only TCP connections without TLS with a remote entity are allowed |
| TLS permitted | TCP connections with and without TLS with a remote entity are allowed |
| TLS required | Only TCP connections with TLS with a remote entity are allowed |

# Host TLS Behavior

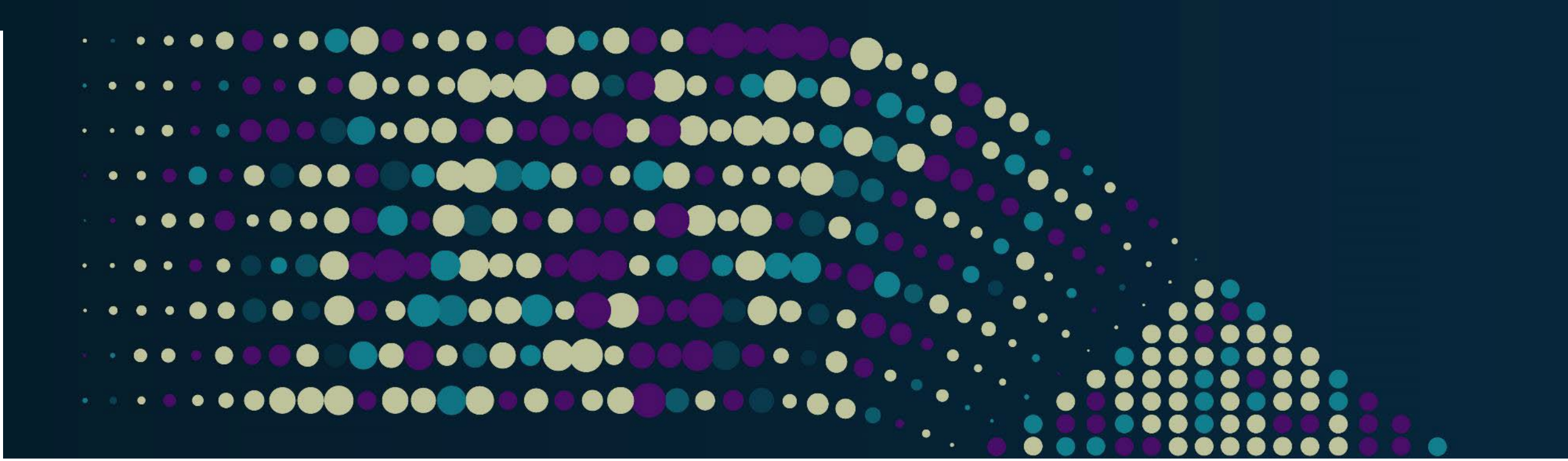
| Host Usage Configuration | Action |
|---|---|
| TLS disabled | Do not initiate TCP connections with TLS. |
| TLS permitted | If the SECTYPE field in the TSAS field in the discovery log entry for the remote entity is not cleared to zero, then initiate TCP connections with TLS, irrespective of the value of the TSC field in that discovery log entry. If establishing any TCP connection with TLS fails and the TSC field in that discovery log entry is not set to 01b (i.e., Required), the host may fall back to initiate TCP connections without TLS.<br>If the SECTYPE field in the TSAS field in the discovery log entry for the remote entity is cleared to zero and the TSC field is not set to 01b (i.e., Required), then initiate TCP connections without TLS. If the SECTYPE field in the TSAS field in the discovery log entry for the remote entity is cleared to zero and the TSC field is set to 01b (i.e., Required), then that discovery log entry is inconsistent and TCP connections without TLS may or may not be initiated.<br>If no discovery log entry has been retrieved for the remote entity, then TCP connections with or without TLS may be initiated. |
| TLS required | Initiate TCP connections with TLS. |

# NVN Subsystem TLS Behavior

| Subsystem Usage Configuration | Action |
|---|---|
| TLS disabled | Close the TCP connection if a TLS handshake is initiated upon completion of the TCP handshake |
| TLS permitted | Continue all TCP connections whether or not a TLS handshake is initiated upon completion of the TCP handshake |
| TLS required | Close the TCP connection if a TLS handshake is not initiated upon completion of the TCP handshake |



# Conclusion

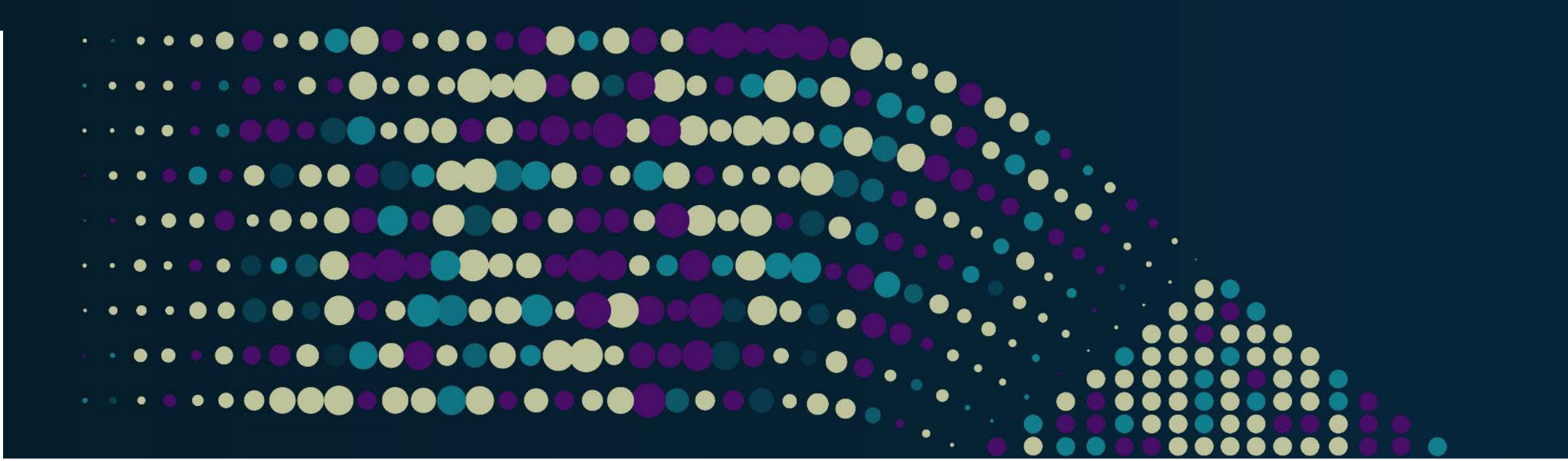
## TP 8018: Updates to NVMe/TCP with TLS

- Yes, NVMe/TCP-TLS can now be used!

## TP 8025: Usage Configuration of NVMe/TCP Security

- Yes, NVMe over IP security usage can now be consistently configured!



# Please take a moment to rate this session

Your feedback is important to us