LEGAL NOTICE:

This Technical Proposal is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS: Members of NVM Express, Inc. have the right to use and implement this Technical Proposal subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2008 to 2023 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "**AS IS**" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVM EXPRESS, INC. (ALONG WITH THE CONTRIBUTORS TO THIS DOCUMENT) HEREBY DISCLAIM ALL REPRESENTATIONS, WARRANTIES AND/OR COVENANTS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, VALIDITY, AND/OR NONINFRINGEMENT.

All product names, trademarks, registered trademarks, and/or servicemarks may be claimed as the property of their respective owners.

The NVM Express® design mark is a registered trademark of NVM Express, Inc.

NVM Express Workgroup
c/o VTM, Inc.
3855 SW 153rd Drive
Beaverton, OR 97003
USA
info@nvmexpress.org

NVM Express® Technical Proposal (TP)

Technical Proposal ID	**8018 - NVMe/TCP – TLS updates**
Revision Date	**2023.09.27**
Builds on Specification(s)	**NVM Express TCP Transport Specification 1.0c** **NVM Express Base Specification 2.0c**
References	**TP 8025 Security Configurations**

Technical Proposal Author(s)

Name	**Company**
Claudio DeSanti, David Black	Dell
Costa Sapuntzakis	Pure
Hannes Reinecke	SUSE
Murali Rajagopal	VMware
Fred Knight	NetApp
Curtis Ballard	HPE

Technical Proposal Overview

Redesign TLS concatenation to DH-HMAC-CHAP:

- Disconnect initial TCP connection after generating TLS PSK, use that TLS PSK on new connections.
- Use new format of TLS PSK identity that uniquely identifies TLS PSK. Old format is obsoleted.
- Only allow TLS PSK generation on Admin Queue connections, prohibit on I/O Queue connections.
- Distinguish generation of a new TLS PSK from replacement of an existing TLS PSK.

TLS requirement updates for NVMe/TCP:

- Forward secrecy: Remove requirement for PSK-only authentication. PSK with ECDHE remains required. Change mandatory DH group for forward secrecy to p384 curve.
- Channel binding: Require NQNs in TLS PSK identity and Connect command to match.
- TLS handshake protection: Enforce existing restrictions that sending any other traffic when a TLS handshake is expected or in progress is an error that closes the TCP connection and that sending unencrypted traffic after a successful TLS handshake is also an error that closes the TCP connection.
- TLS functionality limitation or discouragement:
    - Strongly discourage TLS 1.2 in favor of TLS 1.3 for improved security.
    - Prohibit Application Layer Protocol Negotiation (ALPN) as it is inapplicable to NVMe/TCP.
    - Strongly discourage TLS 1.3 session resumption.
- State that some environments may require stronger crypto than minimum implementation requirements, e.g., environments that follow CNSA 1.0 implementation guidance.
- Specify error if TLS PSK generation attempted with NULL DH group (because that doesn't work).

Revision History

Revision Date	Change Description
2023.01.24	Initial draft
2023.01.31	Initial edits for items A (remove PSK-only requirement) and B (remove requirement for finite field DHE group. Add comments on the nature of the PSK Identity problem (item C).
2023.02.06	Initial partial edits for item D (check NQNs in Connect against TLS PSK identity). Add note indicating where item E (don't nest TLS in TLS) changes will be located in the base specification.
2023.02.08	Complete initial edits for item D (check NQNs in TLS PSK identity for TLS concatenation against the Connect command). Add initial edits for items E (don't nest TLS in TLS) and F (No NVMe traffic during TLS handshake)
2023.02.09	Simplify text for item E (don't nest TLS in TLS). Additional minor edits.
2023.02.10	TLS requirements changes in Section 3.6.1.1
2023.02.14	Updates based on input from Costa Sapuntzakis: • Add error cases for use of NULL DH group with TLS concatenation • Fix text errors in which NQNs to compare. Minor edits from FMDS discussion.
2023.02.17	Add Base spec error descriptions for data sent before secure channel handshake and a comment on the choice of error handling approach to be made in the TCP spec.
2023.02.21	Updates from FMDS discussion: • Accepted most of the changes made to date. • Added comments on rewriting/expanding error cases that cause TLS 1.3 handshake to be aborted or abandoned. • Additional minor edits.
2023.02.28	Change TLS handshake abort text to use passive voice to avoid using "the controller" as the active entity.
2023.03.08	Add remaining error cases for sending data instead of performing TLS 1.3 handshake. Split up section 3.6.1.6 to include subsections on channel binding and sending prohibited data during or instead of the TLS 1.3 handshake. Add note for discussion based on review of RFC 9325.
2023.03.14	Updates from FMDS discussion: • Accept most previous changes, including 2023.03.08 changes • Strengthen prohibition of processing unexpected data. • Add follow-up comments to ensure that non-use of TLS is supported. • Additional minor edits and comments for follow-up.
2023.03.23	Complete review of RFC 9325 recommendations for TLS 1.3 usage by adding prohibition of Application-Layer Protocol Negotiation (ALPN). Generalize text on terminating TLS with an error alert to allow any appropriate error alert. Add text to discourage TLS session resumption. Reviewed RFC 9257 (Guidance for External Pre-Shared Key (PSK) Usage in TLS) – no NVMe/TCP changes appear to be appropriate. Reviewed RFC 9258 (Importing External Pre-Shared Keys (PSKs) for TLS 1.3) – initial proposal is not to adopt RFC 9258.
2023.03.24	TLS concatenation updated to new mechanism that disconnects after generating a TLS PSK and reconnects using that PSK to start TLS. Additional minor edits.
2023.04.04	Updated the PSK Use section. Other minor updates.
2023.04.11	Added TLS 1.2 considerations, specified PSK priorities
2023.04.13	Clarify meaning of TLS concatenation in section 3.6.1.6 of TCP. Clean up Word's automatic figure renumbering. Additional minor edits.
2023.04.17	Add CNSA-related text on crypto algorithms. No text is proposed on comparable strengths of different crypto algorithms (e.g., to compare strengths of secure hash and encryption algorithms)

2023.04.18	SECTYPE field clarifications
2023.04.25	Clarification from Fred Knight: Post-Quantum Cryptography is out of scope. Updated the TLS PSK derivation, minor clarification for TLS concatenation
2023.05.16	Added CRC clarifications, minor phase 3 updates.
2023.05.19	Add TP Overview and Description of Changes material. Add references for two obsolete enumerated values.
2023.05.22	Correct references to controller NQNs (comment from James Smart) Clarify that it is ok to use a TLS connection to replace a TLS PSK.
2023.05.23	Ready for phase 3 exit
2023.05.25	Approved for 30-day member review
2023.07.06	Resolved editorial member review comments from Murali Rajagopal and Mike Allison
2023.07.11	FMDS discussion: Minor edits
2023.07.13	Clean version for integration
2023.09.27	Integrated

Description for Changes Document for NVM Express® TCP Transport Specification 1.0c

New Features/Feature Enhancements/Required Changes:

- TLS concatenation redesign (required if TLS concatenation supported):
  - **New requirement / incompatible change**
    - TLS PSK generation algorithm and TLS PSK identity format replaced with new versions to remove ambiguity. TLS PSK identity now uniquely identifies TLS PSK even if multiple TLS PSKs have been generated by the same host and NVM subsystem pair. The prior algorithm and identity format resulted in ambiguity where multiple TLS PSKs could have the same TLS PSK identity.
  - **New requirement / incompatible change**
    - Restrict TLS PSK generation to Admin Queue connections. A generated TLS PSK may be used on I/O Queue connections and other Admin Queue connections. This simplifies implementations.
  - **New requirement / incompatible change**
    - Distinguish generation of a new TLS PSK from replacement of an existing TLS PSK to support TLS PSK rollover.

  - References
    - Technical Proposal 8018: NVMe/TCP – TLS updates

- TLS requirement updates (required):
  - Add a statement that external requirements (e.g., CNSA 1.0) may mandate stronger crypto algorithms than those specified here.
  - Clarify that CRC representation for PSK Interchange format is little-endian.
  - Prohibit use of TLS Application Layer Protocol Negotiation (ALPN). ALPN usage is not currently possible.

  - **New requirement / incompatible change**
    - TLS version: Strongly discourage use of TLS 1.2 in favor of TLS 1.3 and prohibit use of same TLS PSK with both TLS 1.2 and TLS 1.3.
  - **New requirement / incompatible change**
    - Authentication: Remove requirement for PSK-only authentication. (EC)DHE authentication continues to be required.
  - **New requirement / incompatible change**
    - Forward Secrecy: Update (EC)DHE requirement to require ECDHE and require support of the secp384r1 curve for ECDHE. Remove requirement for finite field DHE and remove related requirement for support of the ffdhe3072 DH group.
  - **New requirement / incompatible change**
    - Prohibit Nesting: Prohibit nesting a TLS session established for NVMe/TCP inside another TLS session established for NVMe/TCP.
  - **New requirement / incompatible change**
    - Prohibit ALPN: Prohibit use of TLS Application Layer Protocol Negotiation (ALPN). ALPN usage is not currently possible.
  - **New requirement / incompatible change**
    - Avoid Session Resumption: Strongly discourage use of TLS session resumption with NVMe/TCP and state that support for TLS session resumption is not required.
  - **New requirement / incompatible change**
    - TLS Channel Binding: Host and NVM Subsystem NQNs must match between TLS PSK identity and Connect command. If the NQNs do not match, the Connect command is aborted with a status code of Connect Invalid Parameters.

- **New requirement / incompatible change**
  - Prohibit unexpected data: In accordance with RFC 8446, prohibit all unexpected data (including NVMe traffic) before and during a TLS handshake and unexpected unencrypted data after a TLS handshake. If this occurs, a TLS alert is delivered (if feasible) and the TCP connection always closes.

- References
  - Technical Proposal 8018: NVMe/TCP – TLS updates

Description for Changes Document for NVM Express Base Specification 2.0c

New Features/Feature Enhancements/Required Changes:

- TLS concatenation redesign (required if TLS concatenation supported):
  - **New requirement / incompatible change**
    - Replace TLS PSK concatenation mechanism that proceeded from DH-HMAC-CHAP to TLS on same TCP connection with a new mechanism that disconnects TCP connection after TLS PSK generation and uses TLS PSK on new connections. This replacement improves security by applying TLS security to the Connect command on any connection that is used for NVMe commands and prevents common implementation flaws that could allow NVMe commands to bypass TLS on the same connection.
  - **New requirement / incompatible change**
    - Restrict TLS PSK generation to Admin Queue connections. A generated TLS PSK may be used on I/O Queue connections and other Admin Queue connections. This simplifies implementations.
  - **New requirement / incompatible change**
    - Distinguish generation of a new TLS PSK from replacement of an existing TLS PSK to support TLS PSK rollover.

  - References
    - Technical Proposal 8018: NVMe/TCP – TLS updates

- TLS requirement updates (required):
  - Add statements that external requirements (e.g., CNSA 1.0) may mandate stronger crypto algorithms than those specified here.
  - Clarify that CRC representation for PSK Interchange format is little-endian.

  - **New requirement / incompatible change**
    - Prohibit Nesting: Prohibit nesting a secure channel established for NVMe-oF inside another secure channel established for NVMe-oF.
  - **New requirement / incompatible change**
    - NULL DH group restriction: Prohibit use NULL DH group with secure channel concatenation (as that group does not support key exchange) and specify authentication failure that results if this is attempted.

  - References
    - Technical Proposal 8018: NVMe/TCP – TLS updates

Markup Conventions:

Black:	Unchanged (however, hot links are removed)
~~Red Strikethrough~~:	Deleted
Blue:	New
Blue Highlighted:	TBD values, anchors, and links to be inserted in new text.
<Green Bracketed>:	Notes to editor

Description of Specification Changes for NVM Express® TCP Transport Specification 1.0c

Modify section 1.5 (References) by adding the following references in the appropriate location:

CNSA 1.0, "USE OF PUBLIC STANDARDS FOR SECURE INFORMATION SHARING", CNSSP 15 ANNEX B "NSA-APPROVED COMMERCIAL NATIONAL SECURITY ALGORITHM (CNSA) SUITE", 20 October 2016. Available from https://www.cnss.gov/CNSS/issuances/Policies.cfm.

RFC 7301, S. Friedl, A. Popov, A. Langley, E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", July 2014. Available from https://www.ietf.org/rfc.html.

Modify section 3.1 (Setup and Initialization) as shown below:

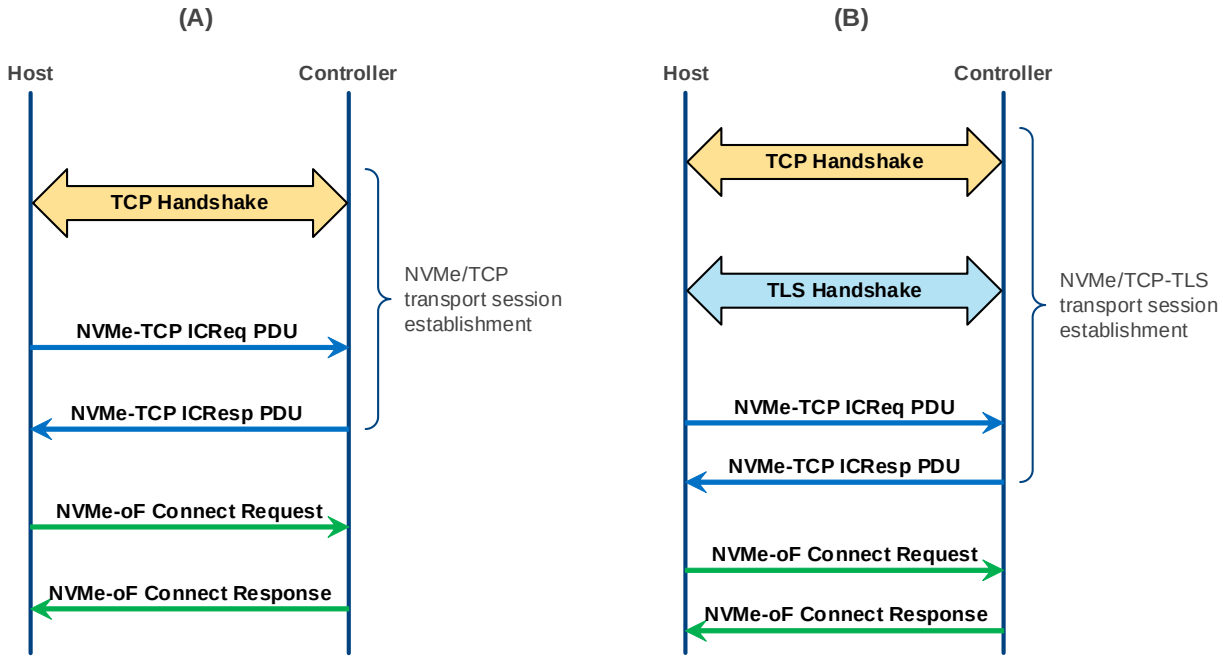
3.1 Setup & Initialization

Figure 7 illustrates the process used to establish an NVMe/TCP connection, with or without TLS. To establish an NVMe/TCP connection without TLS, ~~T~~the first step is to establish a TCP connection between a host and a controller, as shown in Figure 7 (A). To establish an NVMe/TCP connection with TLS (i.e., an NVMe/TCP-TLS connection), the first step is to establish a TCP connection between a host and a controller followed by a TLS handshake, as shown in Figure 7 (B). A controller acts as the passive side of the TCP connection and is set to "listen" for host-initiated TCP connection establishment requests.

Once a TCP or TLS connection has been established, the host sends an Initialize Connection Request (ICReq) PDU to the controller. When a controller receives an ICReq PDU, that controller responds with an Initialize Connection Response (ICResp) PDU. The exchange is used to both establish a connection and exchange connection configuration parameters. When a connection is established, the host and controller are ready to exchange capsules and command data. The first capsule exchange is the NVMe-oF Connect request/response sequence.

<Replace entire Figure 7 with new Figure 7 below>

Figure 7: NVMe/TCP Queue Establishment Sequence



Modify section 3.6.1 (Transport Layer Security) as shown below:

3.6 Transport Specific Content

3.6.1 Transport Layer Security

This section describes the Transport Layer Security (TLS) requirements for the TCP transport. Requirements associated with Post-Quantum Cryptography are outside the scope of this specification.

3.6.1.1 Transport Specific Address Subtype: TLS

As specified in section 3.1.1, a Discovery Log Entry for the TCP transport includes a Transport Specific Address Subtype (TSAS) field. The TSAS field for the TCP transport is ~~SECTYPE field~~ defined in Figure 17 and contains a SECTYPE field that describes whether TLS is supported. TLS implementation is optional for NVMe/TCP.

Figure 17: Transport Specific Address Subtype Definition for NVMe/TCP Transport

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Security Type (SECTYPE): Specifies the type of security ~~used~~ supported by the NVMe/TCP port. ~~If~~ The SECTYPE ~~is a~~ value of 0h (i.e., No Security) specifies that TLS is not supported. ~~, then the host shall set up a normal TCP connection.~~ <Note: TP 8025 specifies comprehensive TLS usage requirements.>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>No Security</td></tr>
<tr><td>01</td><td>Transport Layer Security (TLS) version 1.2 (refer to the NVMe over Fabrics Specification). TLS version 1.2 should not be used with NVMe/TCP.</td></tr>
<tr><td>02</td><td>Transport Layer Security (TLS) version 1.3 (refer to RFC 8446) or a subsequent version. The TLS protocol negotiates the version and cipher suite for each TCP connection.</td></tr>
<tr><td>255:03</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>255:01</td><td>Reserved</td></tr>
</table>

TLS protocol versions prior to 1.2 shall not be used with NVMe/TCP (refer to section 3.1.1 of RFC 8996). All versions of SSL, the predecessor protocol to TLS, shall not be used with NVMe/TCP. NVMe/TCP implementations that are compliant with version 1.0 of the NVMe TCP Transport Specification and that support TLS shall support TLS 1.3 (refer to RFC 8446).

3.6.1.2 Mandatory and Recommended Cipher Suites

TLS for NVMe/TCP is based on pre-shared key (PSK) authentication. NVMe/TCP implementations that support TLS 1.3 shall support the TLS_AES_128_GCM_SHA256 {13h, 01h} cipher suite and should support the TLS_AES_256_GCM_SHA384 {13h, 02h} cipher suite. Implementation and use of the TLS_AES_256_GCM_SHA384 cipher suite may be necessary to meet requirements of security policies that are not defined by NVM Express (e.g., CNSA 1.0).

Implementations shall support disabling individual cipher suites. The methods for disabling individual cipher suites are outside the scope of this specification.

For authentication and key exchange, implementations shall support ~~(refer to section 2 of RFC 8446):~~

- ~~PSK-only authentication; and~~
- PSK with (EC)DHE (refer to RFC 8446).

<Editor's Note: Remove itemization as part of incorporation into .NEXT file>

PSK with (EC)DHE protects encrypted traffic against compromise of the pre-shared key. Implementations that support PSK-only authentication shall support disabling PSK-only authentication. The method for disabling PSK-only authentication is outside the scope of this specification.

The DH exponentials used in an ephemeral DH exchange should not be reused (refer to section 2.12 of RFC 7296 for guidance on DH exponential reuse). Implementations shall not use a DH exponential for multiple protocols (e.g., use the same DH exponential for DH-HMAC-CHAP and TLS).

NVMe/TCP TLS 1.3 implementations shall ~~support the ffdhe3072 group for PSK with DHE and should~~ support the secp384r1 group (refer to section 4.2.7 of RFC 8446) for PSK with ECDHE. Implementations shall support restricting the DH and ECDH groups offered and accepted. The method for restricting the DH and ECDH groups offered and accepted is outside the scope of this specification.

If TLS 1.2 is supported in addition to TLS 1.3, then using the same PSK with both TLS 1.2 and TLS 1.3 is prohibited.

3.6.1.3 TLS PSK and PSK Identity Derivation

This section uses the following terminology:

- **Configured PSK:** the PSK provided via an administrative interface of the NVMe/TCP entity. The method for configuring a PSK is outside the scope of this specification;
- **Retained PSK:** the PSK stored by the NVMe/TCP entity for use with TLS;
- **Generated PSK:** the PSK generated by an NVMe authentication protocol (e.g., DH-HMAC-CHAP, refer to the DH-HMAC-CHAP Protocol section in the NVM Express Base Specification); and
- **TLS PSK:** the PSK used by the TLS protocol.

The configured PSK is configured on both involved entities (i.e., host and NVM subsystem). NVM subsystems should support the ability to use a different configured PSK with each host. Hosts should support the ability to use a different configured PSK with each NVM subsystem.

The retained PSK is derived from the configured PSK (refer to section 3.6.1.4). The configured PSK shall be destroyed as soon as the retained PSK is generated and stored. Each NVMe/TCP entity shall support:

- transforming the configured PSK into a retained PSK before it is stored by the NVMe/TCP entity for repeated use with another NVMe/TCP entity; and
- using the configured PSK as a retained PSK.

The method to derive a retained PSK from a configured PSK shall be using the HKDF-Extract and HKDF-Expand-Label operations (refer to RFC 5869 and RFC 8446):

1. PRK = HKDF-Extract(0, Configured PSK); and
2. Retained PSK = HKDF-Expand-Label(PRK, “HostNQN”, NQN_h, Length(Configured PSK)),

where NQN_h in the NQN of the host. The hash function used with HKDF shall be the one specified in the PSK interchange format (refer to section 3.6.1.4). This transform requires that the NVM subsystem knows the NQN of the host with which the configured PSK is used.

The retained PSK or the generated PSK is used to derive the TLS PSK and the related PSK identity that are associated with a TLS 1.3 cipher suite hash function. The result is a {TLS PSK, PSK Identity, Hash} tuple for use with TLS 1.3.

In TLS 1.3 each PSK is identified by the client using a PSK identity. Each PSK is also associated with one hash function that shall be the same as the hash function of the selected cipher suite. For example, the cipher suites TLS_AES_128_GCM_SHA256 and TLS_AES_256_GCM_SHA384 use the SHA-256 and SHA-384 hash functions respectively. A TLS client that offers both cipher suites shall offer two PSKs with different identities, different hash functions, and different key material.

A TLS 1.3 client implementation that only supports sending a single PSK identity during connection setup may be required to connect multiple times in order to negotiate cipher suites with different hash functions. In this case, the client implementation should connect first with the most preferred PSK. For the cipher suites that are supported, the PSKs should be used in the following order of the associated cipher suite:

1. TLS_AES_256_GCM_SHA384; and
2. TLS_AES_128_GCM_SHA256.

Some TLS 1.3 server implementations are only able to validate one PSK at a time in the order that they are listed in the TLS 1.3 pre_shared_key extension. As a result, TLS 1.3 client implementations should order their offered PSKs from most ~~desirable~~ preferred to least ~~desirable~~preferred. For the cipher suites that are supported, the PSKs should be listed in the following order of the associated cipher suite:

1. TLS_AES_256_GCM_SHA384; and
2. TLS_AES_128_GCM_SHA256.

The TLS 1.3 PSK identity used with NVMe/TCP is generated from the NQNs of the host and the NVM subsystem that contains the controller and from a digest of the PSK to which the identity is associated. The PSK identity is a UTF-~~-~~8 string constructed as an in-order concatenation of the following elements.

1. A 4-character format specifier “NVMe” in ~~utf~~UTF-~~-~~8 encoding;
2. A one-character TLS protocol version indicator, as shown in Figure NEW.2;

Figure NEW.2: TLS protocol version indicator

Value	Description
‘0’ (i.e., U+0030h)	~~indicates TLS 1.3~~ Obsolete (refer to NVM Express TCP Transport Specification 1.0)
‘1’ (i.e., U+0031h)	Indicates TLS 1.3 with PSK digest in the PSK identity
All other values	Reserved

3. A one-character PSK type indicator, specifying the used PSK, as shown in Figure NEW.3;

Figure NEW.3: PSK type indicator

Value	Description
‘R’ (i.e., U+0052h)	Indicates ~~the~~ a retained PSK
‘G’ (i.e., U+0047h)	Indicates ~~the~~ a generated PSK
All other values	Reserved

4. A two-character~~s~~ hash specifier, specifying the hash function of the cipher suite associated with this PSK identity, as shown in Figure NEW.4;

Figure NEW.4: Hash function indicator

Value	Description
“01”	Indicates SHA-256 (e.g., for the TLS_AES_128_GCM_SHA256 cipher suite)
“02”	Indicates SHA-384 (e.g., for the TLS_AES_256_GCM_SHA384 cipher suite)
All other values	Reserved

5. A space character (i.e., U+0020h);
6. The NQN of the host (i.e., NQN_h);
7. A space character (i.e., U+0020h);
8. The NQN of the NVM subsystem that contains the controller (i.e., NQN_c);
9. A space character (i.e., U+0020h);
10. The PSK digest; and
11. A null character (i.e., U+0000h).

The PSK digest shall be computed by encoding in Base64 (refer to RFC 4648) the result of the application of the HMAC function using the hash function specified in element 4 of the above in-order concatenation with the PSK as the HMAC key to the concatenation of:

- the NQN of the host (i.e., NQN_h) not including the null terminator;
- a space character;
- the NQN of the NVM subsystem (i.e., NQN_c) not including the null terminator;
- a space character; and
- the seventeen ASCII characters “NVMe-over-Fabrics”,

(i.e., <PSK digest> = Base64(HMAC(PSK, NQN_h || " " || NQN_c || " " || "NVMe-over-Fabrics"))).

The length of the PSK digest depends on the hash function used to compute it as follows:

- If the SHA-256 hash function is used, the resulting PSK digest is 44 characters long; or
- If the SHA-384 hash function is used, the resulting PSK digest is 64 characters long.

For example, host NQN "nqn.2014-08.org.nvmexpress:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6" and subsystem NQN "nqn.2014-08.org.nvmexpress:uuid:36ebf5a9-1df9-47b3-a6d0-e9ba32e428a2" with the SHA-256 hash function and the retained PSK generate the following PSK identity:

"NVMe~~0~~1R01 nqn.2014-08.org.nvmexpress:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 nqn.2014-08.org.nvmexpress:uuid:36ebf5a9-1df9-47b3-a6d0-e9ba32e428a2 <PSK digest>"

The TLS PSK shall be derived as follows from an input PSK (i.e., either a retained PSK or a generated PSK) and a PSK identity using the HKDF-Extract and HKDF-Expand-Label operations (refer to RFC 5869 and RFC 8446) where the hash function is the one specified by the hash specifier of the PSK identity:

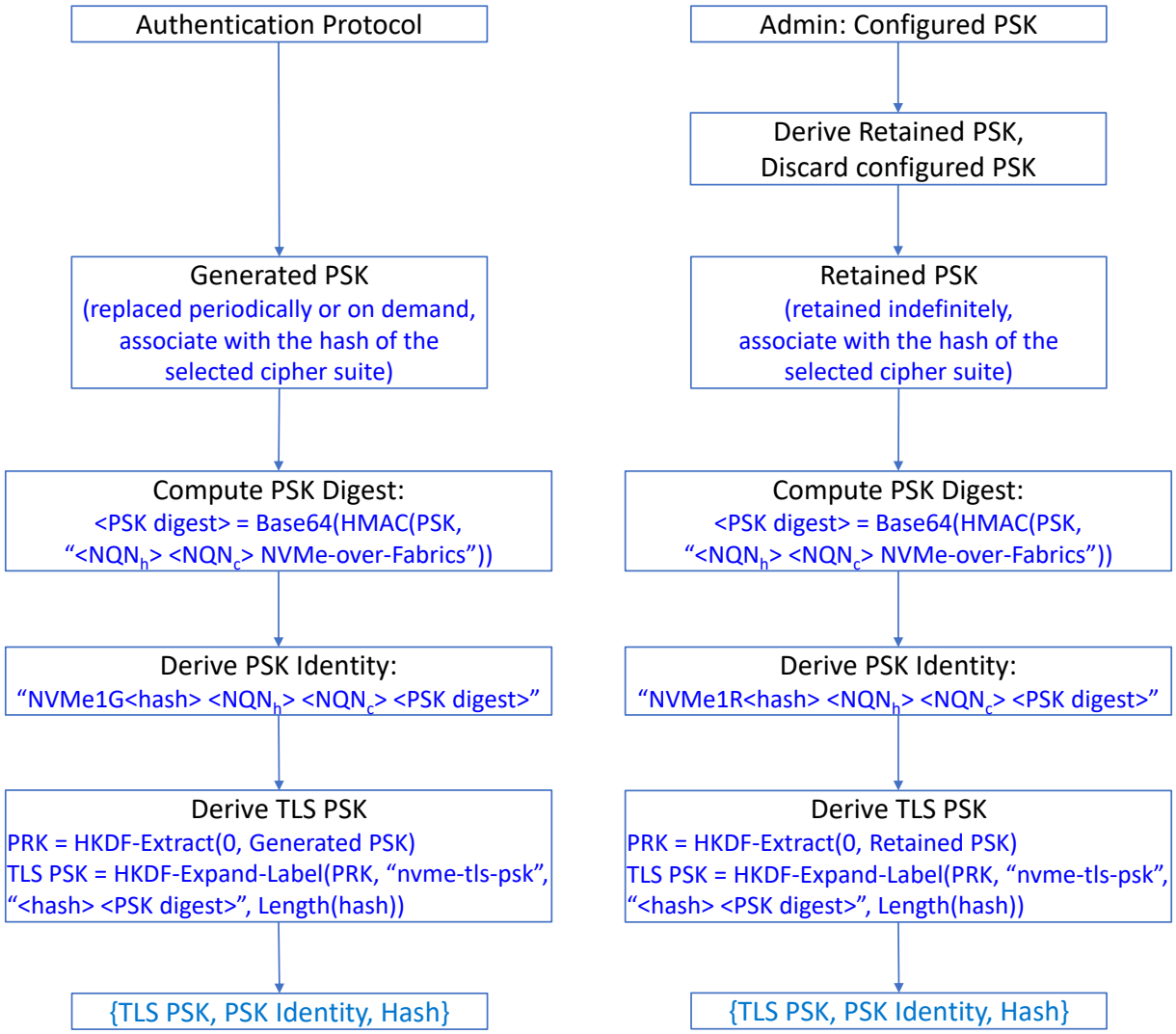
1. PRK = HKDF-Extract(0, Input PSK); and
2. TLS PSK = HKDF-Expand-Label(PRK, "nvme-tls-psk", ~~PskIdentity~~Context, L),

where ~~PskIdentity~~Context is the hash identifier indicated in the PSK identity concatenated to a space character and to the Base64 PSK digest (i.e., "<hash> <PSK digest>") and L is the output size in bytes of the hash function (i.e., 32 for SHA-256 and 48 for SHA-384).

The full process to derive the {TLS PSK, PSK Identity, Hash} tuple is shown in Figure 18 for both configured PSK and generated PSK. This process is performed for each supported TLS 1.3 cipher suite hash function.

<Replace entire Figure 18 with new Figure 18 below>

Figure 18: {TLS PSK, TLS Identity, Hash} Tuple Derivation



The TLS PSK derivation specified in the NVM Express TCP Transport Specification Revision 1.0 is based on a PSK identity having "0" as TLS protocol version indicator and is not compatible with the process specified in this specification. The NVM Express TCP Transport Specification Revision 1.0 specification of the TLS PSK derivation is ambiguous if the associated PSK identity length exceeds 256 characters, hence interoperability problems may arise if the associated PSK identity is longer than 256 characters.

3.6.1.4 PSK ~~Reuse~~Use

If a retained PSK is available for a host and an NVM subsystem, then that ~~A~~ retained PSK and its related PSK Identity ~~is~~ shall be used between ~~a~~ that host and ~~an~~ that NVM subsystem to set up all TLS secure channels for the Admin Queue and all I/O Queues of each controller associated with that host and the use of any generated PSK between that host and that subsystem is prohibited.

If a retained PSK is not available for a host and an NVM subsystem, then one or more generated PSKs shall be used between that host and that NVM subsystem to set up any TLS secure channels.

A generated PSK and its related PSK identity are derived by performing an authentication transaction with the SC_C field in the AUTH_Negotiate message set to NEWTLSPSK (refer to the AUTH_Negotiate Message section of the NVM Express Base Specification) on an Admin Queue over a TCP channel without

TLS. Once that authentication transaction is completed, the NVMe/TCP transport connection of that Admin Queue shall be disconnected (refer to the Secure Channel Concatenation section of the NVM Express Base Specification). The generated {PSK, PSK Identity} pair may be used to set up TLS secure channels for subsequent Admin and I/O queues, as specified in section 3.1.

Once the TLS secure channel for the Admin Queue of an association has been set up with a generated {PSK, PSK Identity} pair, that generated {PSK, PSK Identity} pair should be replaced periodically (e.g., every hour) or on demand by performing a reauthentication with the SC_C field in the AUTH_Negotiate message set to REPLACETLSPSK (refer to the AUTH_Negotiate Message section of the NVM Express Base Specification) over the Admin Queue of that association. The most recently generated PSK, if any, is the generated PSK associated with that Admin Queue.

The TLS secure channels for each I/O Queue of an association shall be set up using the generated PSK that is associated with the Admin Queue of that association. TLS secure channels already set up with a generated PSK shall not be affected by a PSK replacement.

A generated {PSK, PSK Identity} pair may be used to set up the TLS secure channels for Admin Queues of additional associations between the same host and NVM subsystem. In this case that generated {PSK, PSK Identity} pair should be replaced periodically (e.g., every hour) or on demand by performing a reauthentication with the SC_C field in the AUTH_Negotiate message set to REPLACETLSPSK (refer to the AUTH_Negotiate Message section of the NVM Express Base Specification) over the Admin Queue of the association having the smallest Controller ID among the associations using that {PSK, PSK Identity} pair.

The process of generating a PSK on an Admin Queue over a TCP channel without TLS, disconnecting that Admin Queue transport connection, and setting up an Admin Queue over a TLS secure channel established using that generated PSK is called TLS concatenation.

~~A generated PSK used to set up an Admin Queue TLS secure channel after having performed an authentication transaction between a host and a controller (refer to section 6.5.9) may be reused within the lifetime of the generated PSK to set up additional TLS secure channels for I/O Queues on the same controller. Subsequent creation of an I/O Queue on the same controller requires a different PSK, which may be generated by performing another authentication transaction on that I/O Queue. The resulting generated PSK may be reused within the lifetime of the generated PSK to set up additional TLS secure channels for I/O Queues on the same controller.~~

~~A generated PSK shall be discarded when its lifetime expires. The lifetime of a generated PSK shall be ten minutes, unless otherwise configured.~~

~~If a retained PSK exists for an Admin Queue or a I/O Queue associated with a generated PSK, then that generated PSK shall not be reused.~~

A host specifies the PSK identity of the PSK to use to set up a TLS secure channel with a subsystem in the ClientHello message of the TLS handshake. If that PSK, either retained or generated, is not available on the subsystem, then:

- the TLS 1.3 handshake shall be aborted with an unknown_psk_identity alert; and
- the TCP connection shall be closed.

If the host was attempting to set up with a generated PSK the TLS secure channel for an I/O Queue belonging to an existing secured association (i.e., an association whose Admin Queue was set up over a TLS secure channel), then to set up that TLS secure channel the host shall:

- first replace the association's {PSK, PSK Identity} pair by performing a reauthentication with the SC_C field in the AUTH_Negotiate message set to REPLACETLSPSK (refer to the AUTH_Negotiate Message section of the NVM Express Base Specification) over the Admin Queue of that association; and
- then attempt again to set up the TLS secure channel for that I/O Queue.

If the host was attempting to set up with a generated PSK the TLS secure channel for the Admin Queue of a new association, then to set up that TLS secure channel the host shall:

- first perform an authentication transaction with the SC_C field in the AUTH_Negotiate message set to NEWTLSPSK (refer to the AUTH_Negotiate Message section of the NVM Express Base Specification) over an unsecured channel to generate a new {PSK, PSK Identity} pair useable for that new association; and
- then attempt again to set up the TLS secure channel for that Admin Queue.

3.6.1.5 PSK Interchange Format

In order to facilitate provisioning, management, and interchange (e.g., copy & paste in an administrative configuration tool) of PSKs, all NVMe-oF entities shall support the following ASCII representation of configured PSKs:

```
NVMeTLSkey-1:xx:<Base64 encoded string>:
```

Where:

1. ”NVMeTLSkey-1” indicates this is a version 1 representation of a TLS PSK;
2. ‘:’ is used both as a separator and a terminator;
3. xx indicates the hash function to be used to transform the configured PSK in a retained PSK (refer to section 3.6.1.3), encoded as follows:
   - the two ASCII characters “00” indicate no transform (i.e., the configured PSK is used as a retained PSK);
   - the two ASCII characters “01” indicate SHA-256; and
   - the two ASCII characters “02” indicate SHA-384;

   and
4. The Base64 (refer to RFC 4648) string encodes the configured PSK (32 or 48 bytes binary) followed by the CRC-32 (refer to RFC 1952) of the configured PSK (4 bytes binary).

As an example, the 32-byte configured PSK:

`5512DBB6_ 737D0106_ F65975B7_ 73DFB011_ FFC344BC_ F442E2DD_ 6D8BC487_ 0B5D5B03h`

has the CRC-32 `D9BAF45Fh`, that is represented in little endian format (i.e., `5FF4BAD9h`) for concatenation to the configured PSK, resulting in the following: ~~is represented as:~~

```
NVMeTLSkey-1:01:VRLbtnN9AQb2WXW3c9+wEf/DRLz0QuLdbYvEhwtdWwNf9LrZ:
```

when requested to be transformed to a retained PSK with the SHA-256 hash.

When provided with a configured PSK in this format, NVMe-oF entities shall verify the validity of the provided PSK by computing the CRC-32 value of the PSK and checking the computed value with the provided value. If they do not match, then the PSK shall not be used.

3.6.1.6 TLS Implementations and Use Requirements

NVMe/TCP host and subsystem implementations shall not send or use 0-RTT data as it is subject to replay attacks (refer to Appendix E.5 of RFC 8446).

All NVMe/TCP host and subsystem implementations shall be configurable to require that all NVMe/TCP connections use TLS. If a host that supports TLS for NVMe/TCP receives a discovery log entry indicating that the NVM subsystem uses NVMe/TCP and does not support TLS, then the host should nonetheless attempt to establish an NVMe/TCP connection that uses TLS. This requirement applies independent of whether the host is configured to require use of TLS for all NVMe/TCP connections.

NVMe/TCP implementations that support TLS shall support disabling the following parameters, using a method outside the scope of this specification:

- each individual cipher suite;
- PSK-only authentication, if supported;
- each individual DH group; and

- each individual ECDH group.

TLS concatenation is prohibited over a TLS 1.3 session that has been established using a PSK as specified in section 3.6.1. The controller failure response to a host request for such TLS concatenation is specified in the AUTH_Negotiate section of the NVM Express Base Specification. In contrast, replacing a TLS PSK for such a TLS 1.3 session is permitted (refer to section 3.6.1.4).

TLS Application-Layer Protocol Negotiation enables negotiation of the application-level protocol to be used with TLS in situations where multiple application protocols are supported on the same TCP or UDP port (refer to RFC 7301). NVMe/TCP is not expected to be deployed as one of multiple protocols that are supported on the same TCP port and no ALPN Protocol ID has been registered for NVMe/TCP. For these reasons, ALPN shall not be used with TLS for NVMe/TCP.

TLS 1.3 implementations for NVMe/TCP are not required to support TLS session resumption (refer to section 2.2 of RFC 8446), and TLS session resumption should not be used with NVMe/TCP.

3.6.1.6.1 TLS Channel Binding

If a Connect command (refer to the NVM Express Base Specification) is sent over a TLS 1.3 session established using a PSK as specified in section 3.6.1, then the controller shall check whether:

- the NVM Subsystem NVMe Qualified Name (SUBNQN) parameter of the Connect command is the same as the NQN of the controller (i.e., NQN_c) in the identity of that PSK; and
- the Host NVMe Qualified Name (HOSTNQN) parameter of the Connect command is the same as the NQN of the host (i.e., NQN_h) in the identity of that PSK.

If either the SUBNQN check or the HOSTNQN check fails during processing of the Connect command (i.e., the compared NQNs are not the same), then the controller shall abort that Connect command with a status code of Connect Invalid Parameters (refer to the Connect Command and Response section of the NVM Express Base specification).

3.6.1.6.2 Unexpected Data During TLS 1.3 Handshake

A TLS 1.3 handshake is initiated by the host upon completing the TCP handshake at the start of a TCP connection (refer to section 3.1). If the host initiates such a TLS 1.3 handshake, then the host and controller are prohibited from sending any other data instead of initiating or performing the TLS 1.3 handshake, as further specified in this section. This prohibition does not apply to TCP connections that do not use TLS.

A host that initiates the TLS 1.3 handshake at the start of a TCP connection shall not send any other data before initiating the TLS 1.3 handshake. If a controller receives data that is prohibited by this requirement, then the controller shall:

- immediately close the TCP connection without attempting to determine whether the host initiated the TLS 1.3 handshake after sending the prohibited data;
- treat the prohibited data as opaque data; and
- not process that prohibited data (e.g., as if it consisted of one or more NVMe/TCP PDUs).

A controller shall not send any data on a TCP connection until the controller receives data from the host and determines whether that data indicates that the host has initiated a TLS 1.3 handshake. If a host receives data that is prohibited by this requirement, then the host shall:

- abort the TLS 1.3 handshake with an unexpected_message alert if the host has already initiated the TLS 1.3 handshake;
- close the TCP connection regardless of whether or not the host has initiated the TLS 1.3 handshake; and
- treat the prohibited data as opaque data and not process that data (e.g., as if that data consisted of one or more NVMe/TCP PDUs).

A host shall not send any other data (e.g., NVMe/TCP PDUs) while a TLS 1.3 handshake is in progress. If a controller receives any other data (e.g., NVMe/TCP PDUs) while a TLS 1.3 handshake is in progress (i.e., before receiving the TLS 1.3 Finished message), then receiving that data (e.g., NVMe/TCP PDUs) shall

cause the TLS 1.3 handshake to be aborted with an appropriate error alert (e.g., decode_error or unexpected_message), as specified in section 6.2 of RFC 8446, and shall cause the TCP connection to be closed. That received data shall be treated as opaque data and shall not be processed (e.g., as if that data consisted of one or more NVMe/TCP PDUs).

A controller shall not send any other data (e.g., NVMe/TCP PDUs) while a TLS 1.3 handshake is in progress. If a host receives any other data (e.g., NVMe/TCP PDUs) while a TLS 1.3 handshake is in progress (i.e., before receiving the TLS 1.3 Finished message), then receiving that data (e.g., NVMe/TCP PDUs) shall cause the TLS 1.3 handshake to be aborted with an appropriate error alert (e.g., decode_error or unexpected_message), as specified in section 6.2 of RFC 8446, and shall cause the TCP connection to be closed. That received data shall be treated as opaque data and shall not be processed (e.g., as if that data consisted of one or more NVMe/TCP PDUs).

All data sent after completion of the TLS 1.3 handshake (i.e., after sending the TLS 1.3 Finished message) shall be encrypted (refer to section 4.4.4 of RFC 8446). If a host or controller receives unencrypted data after completion of the TLS 1.3 handshake, then receiving that data shall cause the TLS 1.3 session to be aborted with an appropriate alert (e.g., decode_error or unexpected_message), as specified in section 6.2 of RFC 8446, and shall cause the TCP connection to be closed. That unencrypted data shall be treated as opaque data and shall not be processed (e.g., as if that data consisted of one or more NVMe/TCP PDUs).

Description of Specification Changes for NVM Express Base Specification 2.0c

Modify section 1.8 (References) by adding the following reference in the appropriate location:

CNSA 1.0, “USE OF PUBLIC STANDARDS FOR SECURE INFORMATION SHARING”, CNSSP 15 ANNEX B “NSA-APPROVED COMMERCIAL NATIONAL SECURITY ALGORITHM (CNSA) SUITE”, 20 October 2016. Available from https://www.cnss.gov/CNSS/issuances/Policies.cfm.

Modify section 8.13.1 (Fabric Secure Channel) as follows:

8.13.1 Fabric Secure Channel

The Transport Requirements field in the Fabrics Discovery Log Page Entry (refer to Figure 264) indicates whether a fabric secure channel shall be used for an NVMe Transport connection to an NVM subsystem. The secure channel mechanism is specific to the type of fabric.

If establishment of a secure channel fails or a secure channel is not established when required by the controller, the resulting errors are fabric-specific and may not be reported to the NVMe layer on the host. Such errors may result in the controller being inaccessible to the host via the NVMe Transport connection on which the failure to establish a fabric secure channel occurred.

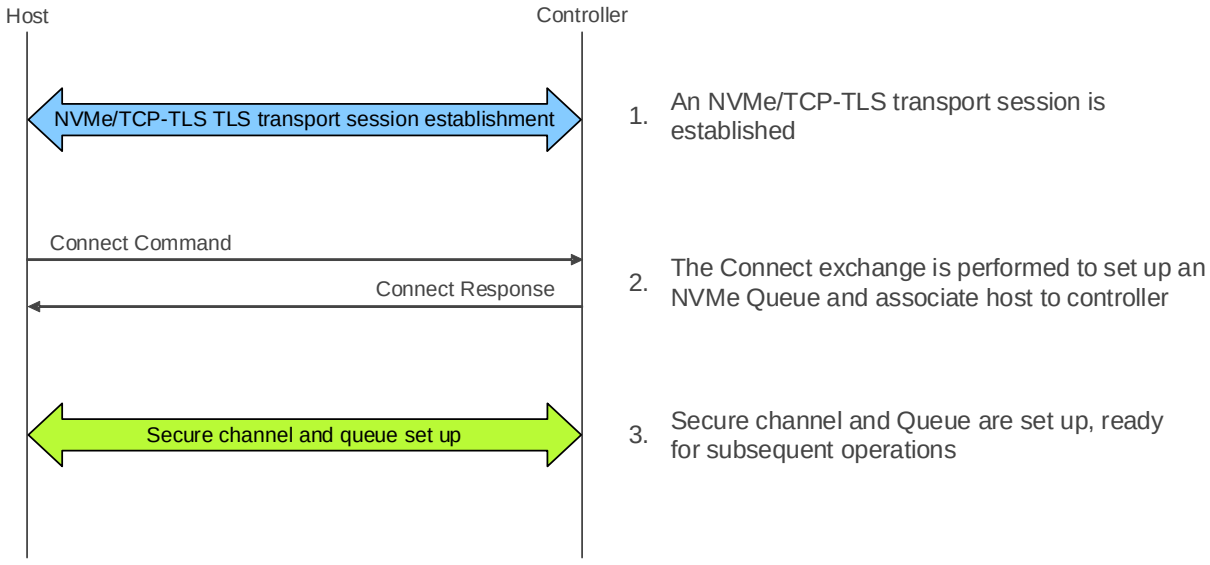
An NVM subsystem that requires use of a fabric secure channel (i.e., as indicated by the TREQ field in the associated Discovery Log Page Entry) shall not allow capsules to be transferred until a secure channel has been established for the NVMe Transport connection.

All Discovery Log Page Entries for an NVM subsystem should report the same value of TREQ to each host. Discovery Log Page Entries for an NVM subsystem may report different values of TREQ to different hosts.

Figure 432 shows an example of secure channel establishment using TLS, the fabric secure channel protocol for NVMe/TCP.

<Replace entire Figure 432 with new Figure 432 below>

Figure 432: Example of TLS secure channel establishment



Modify section 8.13.2 (NVMe In-band Authentication) as follows:

8.13.2 NVMe In-band Authentication

The Authentication and Security Requirements (AUTHREQ) field in the Connect response capsule (refer to Figure 382) indicates whether NVMe in-band authentication is required.

If one or more of the bits in the AUTHREQ field are set to '1', then the controller requires that the host authenticate on that queue in order to proceed with Fabrics, Admin, and I/O commands. Authentication success is defined by the specific security protocol that is used for authentication. If any command other than Connect, Authentication Send, or Authentication Receive is received prior to authentication success, then the controller shall abort the command with Authentication Required status.

If all bits in the AUTHREQ field are cleared to '0', then the controller does not require the host to authenticate, and the NVM subsystem shall not abort any command with a status code value of Authentication Required.

If NVMe in-band authentication succeeds, then any supported commands for the associated queue type may be processed.

The host may initiate a subsequent authentication transaction at any time for reauthentication purposes. Initiating reauthentication shall not invalidate a prior authentication. If the reauthentication transaction concludes with the controller sending an AUTH_Failure1 message (refer to section 8.13.4.2), then the controller shall terminate all commands with a status code of Operation Denied and disconnect the NVMe over Fabrics connection. If the reauthentication transaction concludes with the host sending an AUTH_Failure2 message, then the host shall disconnect the NVMe over Fabrics connection.

The state of an in-progress authentication transaction is soft-state. If the subsequent command in an authentication transaction is not received by the controller within a timeout equal to:

- the Keep Alive Timeout value (refer to Figure 380), if the Keep Alive Timer is enabled; or
- the default Keep Alive Timeout value (i.e., two minutes), if the Keep Alive Timer is disabled;

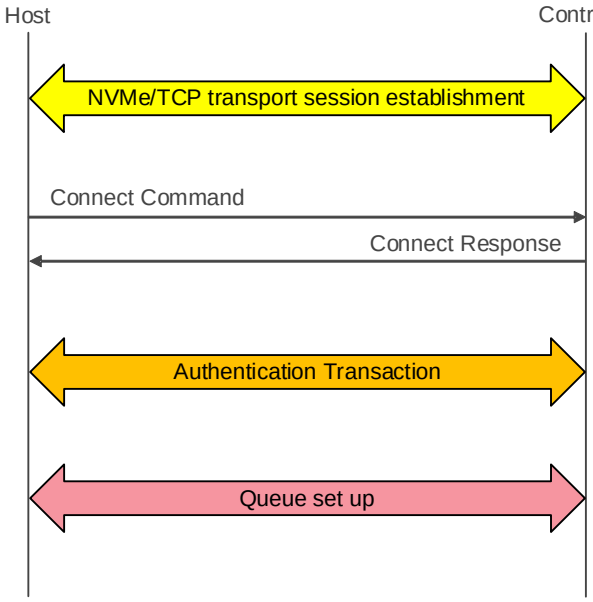
then the authentication transaction has timed out and the controller should discard the authentication transaction state (including the T_ID value, refer to section 8.13.4.1).

For an initial authentication, an authentication transaction timeout should be treated as an authentication failure with termination of the transport connection. For reauthentication, an authentication transaction timeout should not be treated as an authentication failure. Authentication commands used to continue that transaction after an authentication transaction timeout should be aborted with a status code of Command Sequence Error.

Figure 433 shows an example of authentication transaction for NVMe/TCP.

<Replace entire Figure 433 with new Figure 433 below>

Figure 433: Example of authentication transaction for NVMe/TCP



1. An NVMe/TCP transport session is established
2. The Connect exchange is performed to set up an NVMe Queue and associate host to controller
3. The host performs an authentication transaction with the controller to authenticate the end-points
4. Queue is set up, ready for subsequent operations

Modify section 8.13.3 as follows:

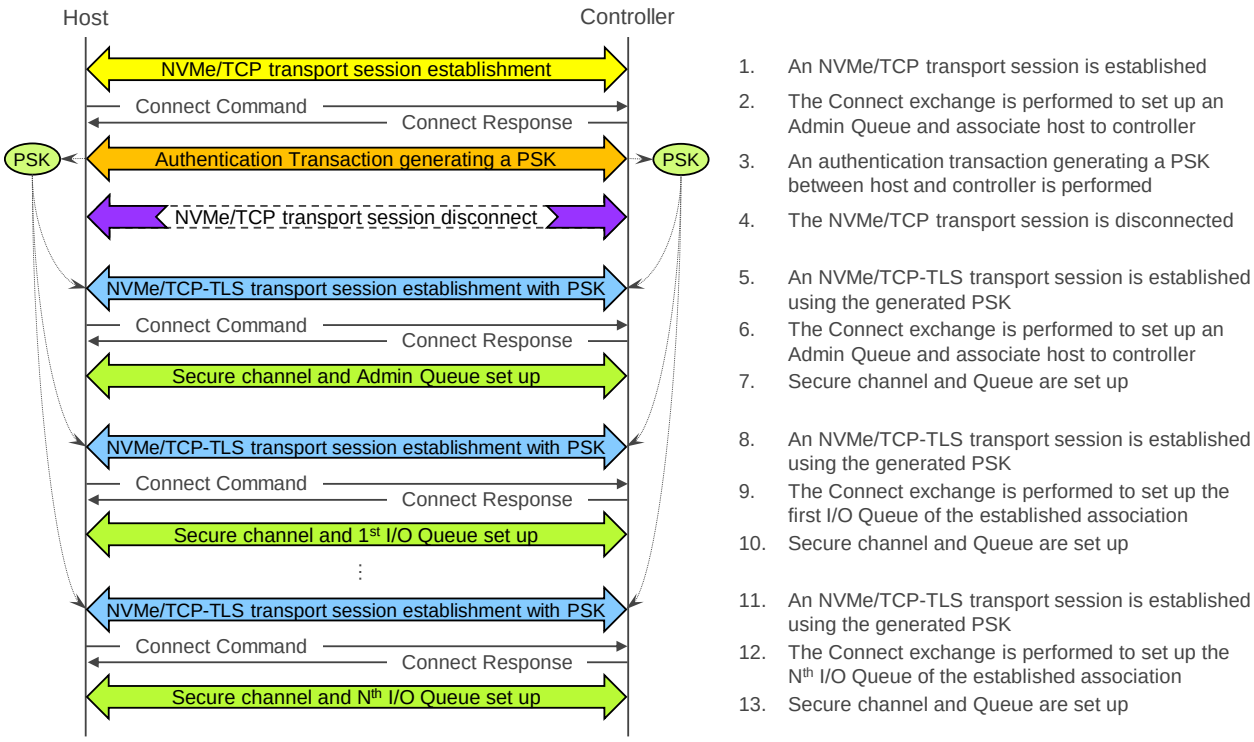
8.13.3 ~~NVMe In-band Authentication Followed by Secure Channel Establishment~~ Secure Channel Concatenation

It is possible to leverage an authentication transaction to generate shared key material to use as pre-shared key (PSK) to establish a secure channel (e.g., with IPsec or TLS). ~~In this case, the PSK generated to set up a secure channel on the Admin Queue may be reused to set up additional secure channels on the I/O queues.~~ This PSK is generated by an authentication transaction on an Admin Queue over an unsecure channel. Once the authentication transaction is completed, that Admin Queue transport connection shall be disconnected by the host. The generated PSK may then be used to set up secure channels for subsequent Admin Queue(s) and I/O Queues.

The process of generating a PSK on an Admin Queue over an insecure channel, disconnecting that Admin Queue transport connection, and setting up an Admin Queue over a secure channel established using that generated PSK is called secure channel concatenation. Figure 436 shows an example of this ~~possibility~~ process for TLS with NVMe/TCP, where steps 1 to 7 show TLS secure channel concatenation for the Admin Queue and steps 8 to 13 show the setup of TLS secure channels for the I/O Queues.

<Replace entire Figure 436 with new Figure 436 below>

Figure 436: Example of TLS secure channel concatenation with NVMe/TCP~~ed to an authentication transaction~~



Secure channel concatenation is prohibited over any secure channel that has been established in compliance with the requirements of an NVMe transport specification (e.g., the NVM Express TCP Transport Specification). The controller response to a host that requests secure channel concatenation in this situation is specified in section 8.13.4.1. In contrast, replacing a PSK for such a secure channel is permitted (refer to section 8.13.4.1).

Modify section 8.13.4.1 (AUTH_Negotiate Message) as shown below:

8.13.4.1 AUTH_Negotiate Message

…

The SC_C field determines if a secure channel concatenation to the authentication transaction is requested and with which secure channel protocol, as shown in Figure 438.

Figure 438: Secure channel protocol identifiers

Value	Description	Transport Applicability
00h	**NOSC:** No secure channel concatenation	n/a
01h	~~Secure channel concatenation with TLS (refer to section 8.13.5.9)~~ Obsolete (refer to NVM Express Base Specification 2.0)	~~TCP~~
02h	**NEWTLSPSK:** Used on an Admin Queue over a TCP channel without TLS to generate a PSK and associated PSK identity. This {PSK, PSK Identity} pair may be used to set up TLS secure channels for subsequent Admin and I/O queues.	TCP
03h	**REPLACETLSPSK:** Used on an Admin Queue over a TLS secure channel to generate a PSK and associated PSK identity. This {PSK, PSK Identity} pair replaces the {PSK, PSK Identity} pair that was used to set up the TLS secure channel over which the authentication transaction is performed.	TCP with TLS
All other values	Reserved	

An authentication transaction with the SC_C field set to NOSC is allowed on any Admin or I/O Queue and does not generate a PSK (i.e., it performs authentication only).

An authentication transaction with the SC_C field set to NEWTLSPSK:

- is allowed on an Admin Queue over a TCP channel without TLS and generates a new PSK;
- is prohibited on an I/O Queue over a TCP channel without TLS; and
- is prohibited on an Admin Queue or I/O queue over a TLS secure channel.

An authentication transaction with the SC_C field set to REPLACETLSPSK:

- is allowed on an Admin Queue over a TLS secure channel and replaces the association's PSK;
- is prohibited on an Admin Queue or I/O queue over a TCP channel without TLS; and
- is prohibited on an I/O Queue over a TLS secure channel.

…

Upon receiving an AUTH_Negotiate message, if the SC_C value indicated by the host:

- does not satisfy the security requirements of the controller (e.g., the host did not request secure channel concatenation, but the controller's security policy requires secure channel concatenation)~~.~~;
- is prohibited for the Admin Queue or I/O Queue via which the AUTH_Negotiate message was received, as specified in this section; or
- requests secure channel concatenation and that value is contained in an AUTH_Negotiate message received over a secure channel established in compliance with the requirements of an NVMe transport specification (e.g., the NVM Express TCP Transport Specification),

then the controller shall:

- reply to the AUTH_Negotiate message with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Secure channel concatenation mismatch'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

…

Modify section 8.13.5.2 (8.13.5.2 DH-HMAC-CHAP Authentication Protocol Descriptor) as follows:

8.13.5.2 DH-HMAC-CHAP Authentication Protocol Descriptor

…

The SHA-256 hash function shall be supported. Use of the SHA-256 hash function may be prohibited by the requirements of security policies that are not defined by NVM Express (e.g., CNSA 1.0 requires use of SHA-384).

...

The 00h identifier indicates that no Diffie-Hellman exchange is performed, which reduces the DH-HMAC-CHAP protocol to the HMAC-CHAP protocol. The 00h identifier shall not be proposed in an AUTH_Negotiate message that requests secure channel concatenation (i.e., with the SC_C field set to a non-zero value).

The 2048-bit DH group and the 3072-bit DH group shall be supported. A mechanism shall be provided to disable (i.e., prohibit) use of the 2048-bit DH group. Use of the 2048-bit DH group may be prohibited by the requirements of security policies that are not defined by NVM Express (e.g., CNSA 1.0 requires use of a 3072-bit or larger DH group).

Upon receiving an AUTH_Negotiate message, if the DHgIDList proposed by the host:

- does not satisfy the security requirements of the controller (e.g., the host proposed only the NULL DH group, but the controller's security policy requires a DH group whose size is 3072-bit or larger)~~,~~; or
- contains the NULL DH group (i.e., identifier 00h) and the AUTH_Negotiate message is requesting secure channel concatenation (i.e., with the SC_C field set to a non-zero value),

then the controller shall:

- reply to the AUTH_Negotiate message with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'DH group not usable'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

…

Modify section 8.13.5.3 (DH-HMAC-CHAP_Challenge Message) as shown below:

8.13.5.3 DH-HMAC-CHAP_Challenge Message

…

DHgID: Shall be set to the DH group identifier (refer to Figure 447) selected for this authentication transaction among those proposed in the DH-HMAC-CHAP protocol descriptor in the AUTH_Negotiate message. The controller shall select a DH group identifier in accord with its applicable policy. If this field is cleared to 0h, the DH portion of the DH-HMAC-CHAP protocol shall not be performed in this authentication transaction. The controller shall not clear this field to 00h if the AUTH_Negotiate message (refer to section 8.13.4.1) for this instance of the DH-HMAC-CHAP protocol requested secure channel concatenation (i.e., the SC_C field in that message was set to a non-zero value).

…

Upon receiving a DH-HMAC-CHAP_Challenge message, if:

- the Hash Length (HL) does not match the value specified in Figure 446 for the selected hash function;

- the Sequence Number (SEQNUM) is cleared to 0h;
- DHgID is cleared to 0h and the AUTH_Negotiate message (refer to section 8.13.4.1) that the host sent for this instance of the DH-HMAC-CHAP protocol requested secure channel concatenation (i.e., the SC_C field in that message is set to a non-zero value);
- DHgID is non-zero and the DH Value Length (DHVLEN) is cleared to 0h; or
- DHgID is non-zero and the DH Value (DHV) is 0, 1, or p-1;

then the host shall:

- reply with an AUTH_Failure2 message having reason code 'Authentication failure' and reason code explanation 'Incorrect payload'; and
- disconnect the NVMe over Fabrics connection.

Modify section 8.13.5.8 (Secret Representation) as follows:

8.13.5.8 Secret Representation

…

As an example, the 32-byte secret:

`89AEB31A 874EAF84 841B4673 6B0DFDF2 BA58D30A A2A545A3 E235A352 1E07594Ch`

has the CRC-32 `A70D69FAh`, that is represented in little endian format (i.e., `FA690DA7h`) for concatenation to the secret, resulting in the following:

~~is represented as:~~ "`DHHC-1:00:ia6zGodOr4SEG0Zzaw398rpY0wqipUWj4jWjUh4HWUz6aQ2n:`"

when intended to be used as a key without transform.

…

Modify section 8.13.5.9 (Generated PSK for TLS) as follows:

8.13.5.9 Generated PSK for TLS

When used with a non-NULL DH exchange, the DH-HMAC-CHAP protocol is able to generate a session key K_S used to generate a pre-shared key (PSK) to establish a secure channel session with the TLS protocol between host and controller for NVMe/TCP. A TLS session is concatenated to an authentication transaction when the SC_C indication is set to ~~01h~~NEWTLSPSK in the AUTH_Negotiate message (refer to section 8.13.4.1). A TLS session ~~should~~ shall not be concatenated to an authentication transaction if the involved host and controller are administratively configured with a PSK for use with each other. In this case, host and controller ~~should~~ shall only ~~just~~ establish a TLS session based on the retained PSK derived from that configured PSK.

The session key K_S shall be computed from the ephemeral DH key (i.e., g^{xy} mod p) generated during the DH-HMAC-CHAP transaction by applying the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message (i.e., $K_S = H(g^{xy} \text{ mod } p)$). The size of the session key K_S is determined by the selected hash function, as shown in Figure 446. Specifically:

- The host computes K_S as the hash of the ephemeral DH key resulting from the combination of the random value y selected by the host with the DH exponential (i.e., g^x mod p) received from the controller (i.e., $K_S = H((g^x \text{ mod } p)^y \text{ mod } p) = H(g^{xy} \text{ mod } p)$).
- The controller computes K_S as the hash of the ephemeral DH key resulting from the combination of the random value x selected by the controller with the DH exponential (i.e., g^y mod p) received from the host (i.e., $K_S = H((g^y \text{ mod } p)^x \text{ mod } p) = H(g^{xy} \text{ mod } p)$).

The generated PSK for TLS shall be computed applying the HMAC function using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message with the session key K_S

as key to the concatenation of the two challenges C_1 and C_2 (i.e., generated PSK = HMAC(K_S, C_1 || C_2)). ~~The generated PSK used to set up a TLS secure channel on the Admin Queue may be reused to set up additional TLS secure channels on the I/O queues (refer to the PSK Reuse section of the NVMe TCP Transport Specification). The lifetime of this generated PSK should be no more than ten minutes; this requires authentication for I/O queues created after this time.~~ This generated PSK should be replaced periodically (e.g., every hour) or on demand by performing a reauthentication on the Admin queue of the association with the appropriate SC_C value (refer to section 8.13.4.1).

The host may request secure channel concatenation with the TLS protocol by setting the SC_C field ~~indication~~ in the AUTH_Negotiate message to ~~01h~~NEWTLSPSK while performing only unidirectional authentication. In this case the host shall still send a challenge value C_2 to the controller and clear the sequence number S_2 to 0h to indicate that controller authentication is not requested.