LEGAL NOTICE:

This NVM Express over Fabrics revision 1.1 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 NVM Express over Fabrics revision 1.1 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: **"© 2007 - 2021 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.

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

NVM Express Technical Proposal for New Feature

Technical Proposal ID	**TP 8011 - TLS 1.3 Profile for NVMe/TCP**
Change Date	**02/15/2021**
Builds on Specification	**NVMe-oF 1.1**
References Ratified TPs	**TP 8006**

Technical Proposal Author(s)

Name	Company
David Black, Claudio DeSanti	Dell/EMC
Constantine Sapuntzakis	Pure Storage

Revision History	
Revision Date	**Change Description**
2020/09/11	Initial import into template / Remove TLS 1.2 support
2020/09/22	Introduce new SECTYPE for TLS 1.3
2020/09/24	Allow TLS 1.2 Should support secp384r1 Various smaller fixes
2020/09/28	Discuss constraints of using same PSK with different hash functions Add hash to use with PSK to PSK identity Add PSK Interchange format Smaller cleanups Should -> shall on parameters PSK identity prefix from C -> NVMe Session tickets should not be mistaken for PSK identities Reflection attacks
2020/10/07	Retained / TLS keys first cut Remove prefix from interchange format CRC
2020/10/13	Differentiate generated from retained TLS keys Add hash specifier to key interchange format TLS master key generation via HKDF Remove processing of NQNs into PSK identity Add protocol version and key type indicators to PSK identity
2020/10/27	Remove reflection attacks language
2020/11/17	Editorial rewrite by cds. Conventions: - Text to add in NVMe-oF 1.1 is in blue - Text to delete in NVMe-oF 1.1 is in ~~strikethrough red~~ - New text worth more review is highlighted in magenta - Text deleted from the previous version is in ~~strikethrough orange~~ - Text added to the previous version is in green
2020/12/01	- Added definition for the keyword “obsolete” - Made TLS_AES_128_GCM_SHA256 mandatory and TLS_AES_256_GCM_SHA384 optional - Clarified how to transform a configured PSK in a retained PSK
2020/12/01 concall	- Editorial fixes
2020/12/15	- Added section on PSK reuse - Miscellaneous clarifications in section 7.4.9.3
2020/12/15 concall	- Allowed TLS 1.2 as per NVMe-oF 1.1 - Editorial fixes
2021/01/28	- Incorporated members’ review editorial comments
2021/02/15	- Integrated into the NVMe-oF Specification, Revision 1.1.

Description of Specification Changes

Add the following definition to section 1.4 (Definitions):

1.4.10 obsolete

Keyword indicating functionality that was defined in a previous version of this specification and that has been removed from this specification.

Modify section 7.4.9 (Transport Specific Address Subtype and Transport Service Identifier) as follows:

7.4.9 Transport Specific Address Subtype and Transport Service Identifier

The Discovery Log Entry includes a Transport Specific Address Subtype (TSAS) field that ~~is defined in Figure 71 for the NVMe/TCP Transport. The TSAS field describes TCP connection properties, such as whether TLS is supported~~ describes TCP connection properties such as whether TLS is used (refer to section 7.4.9.1). The Discovery Log Entry also includes a Transport Service Identifier (TRSVCID) field that describes the TCP port to use (refer to section 7.4.9.7).

~~**7.4.9.1 Mandatory and Recommended Cipher Suites**~~

~~TLS for NVMe/TCP is based on pre-shared key (PSK) cipher suites. NVMe/TCP implementations that implement TLS shall support the TLS_PSK_WITH_AES_128_GCM_SHA256 {00h, A8h} cipher suite (refer to RFC 5487), and NVM subsystems should include that cipher suite in the initial set of cipher suites proposed to a host. In addition, the following cipher suites should be supported (refer to RFC 5487):~~

- ~~TLS_PSK_WITH_AES_256_GCM_SHA384 {00h, A9h} cipher suite;~~
- ~~TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 {00h, AAh} cipher suite; and~~
- ~~TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 {00h, ABh} cipher suite.~~

~~The _AES_128_ and _AES_256_ cipher suites differ in cryptographic strength (e.g., the _AES_128 cipher suites specify the use of 128-bit AES encryption and the _AES_256_ cipher suites specify the use of 256-bit AES encryption). The _DHE_ cipher suites differ from their non-_DHE_ cipher suite counterparts in the addition of an ephemeral Diffie-Hellman (DH) exchange to protect encrypted traffic against compromise of the pre-shared key (refer to section 6.3 of RFC 7525). The DH keys (also called exponents) used in any ephemeral DH exchange:~~

10. ~~shall be at least 2,048 bits in size (refer to section 4.3 of RFC 7525); and~~
11. ~~should not be reused (i.e., used for more than one DH exchange) (refer to section 6.4 of RFC 7525).~~

~~The PSK cipher suite framework is described in RFC 4279. NVMe/TCP uses NQNs to identify hosts and NVM subsystems, specifically, in the TLS handshake for a PSK cipher suite:~~

12. ~~The psk_identity field in the ClientKeyExchange message shall contain the host NQN and the subsystem NQN separated by a space (' '=U+0020h) character as a UTF-8 string, including the terminating null (00h) character.~~

~~The following is an example of the psk_identity field in the ClientKeyExchange message assuming that both the host and the NVM subsystem are using the UUID-based format NVMe Qualified Names:~~

13. ~~nqn.2014-08.org.nvmexpress:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 nqn.2014-08.org.nvmexpress:uuid:36ebf5a9-1df9-47b3-a6d0-e9ba32e428a2.~~

~~For interoperability reasons, NVMe/TCP prohibits identity hints in the TLS 1.2 ServerKeyExchange message. The host is expected to be able to determine which identity and pre-shared key to use with the subsystem based on the NVM subsystem NQN indicated in a corresponding discovery log entry acquired before the client Key Exchange message was sent.~~

7.4.9.1 Transport Specific Address Subtype: TLS

The TSAS SECTYPE field defined in Figure 71 describes whether TLS is supported. TLS implementation is optional for NVMe/TCP.

Figure 71: 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 by the NVMe/TCP port. If SECTYPE is a value of zero (No Security), then the host ~~is~~ shall setup a normal TCP connection.
<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 RFC 5246) or a subsequent version. The TLS protocol negotiates the version and cipher suite for each TCP connection.~~ (refer to NVMe-oF 1.1).</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 7525). All versions of SSL, the predecessor protocol to TLS, shall not be used with NVMe/TCP. NVMe/TCP implementations that are compliant with this version of the specification and that support TLS shall support TLS 1.3 (refer to RFC 8446). ~~For further discussion, refer to section 3.1.1 of RFC 7525. The NVMe/TCP prohibition on versions of TLS prior to 1.2 is stronger than the requirements in RFC 7525 because NVMe/TCP is a new protocol.~~

Editor's note: Refactoring should look at the usage of the words "this version of the specification".

7.4.9.2 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.

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).

PSK with (EC)DHE protects encrypted traffic against compromise of the pre-shared key. Implementations 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). 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.

7.4.9.3 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 section 6.5); and
- **TLS PSK:** the PSK used by the TLS protocol.

Note to the editor: 6.5 is a section of TP 8006.

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 7.4.9.4). 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 5689 and RFC8446):

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 7.4.9.4). 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.

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 to least desirable.

The TLS 1.3 PSK identity used with NVMe/TCP is generated from the NQNs of the host and the controller. 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-8 encoding;
2. A one-character TLS protocol version indicator:
   - ‘0’ (i.e., U+0030h) indicates TLS 1.3;

3. A one-character PSK type indicator, specifying the used PSK:
   - ‘R’ (i.e., U+0052h) indicates the retained PSK;
   - ‘G’ (i.e., U+0047h) indicates the generated PSK;
4. A two-characters hash specifier, specifying the hash function of the cipher suite associated with this PSK identity:
   - “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);
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 controller (i.e., NQN_c); and
9. A null character (i.e., U+0000h).

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:

> “`NVMe0R01 nqn.2014-08.org.nvmexpress:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 nqn.2014-08.org.nvmexpress:uuid:36ebf5a9-1df9-47b3-a6d0-e9ba32e428a2`”

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 5689 and RFC8446) 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, L),

where PskIdentity is the PSK identity 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 TBD+1 for both configured PSK and generated PSK. This process is performed for each supported TLS 1.3 cipher suite hash function.

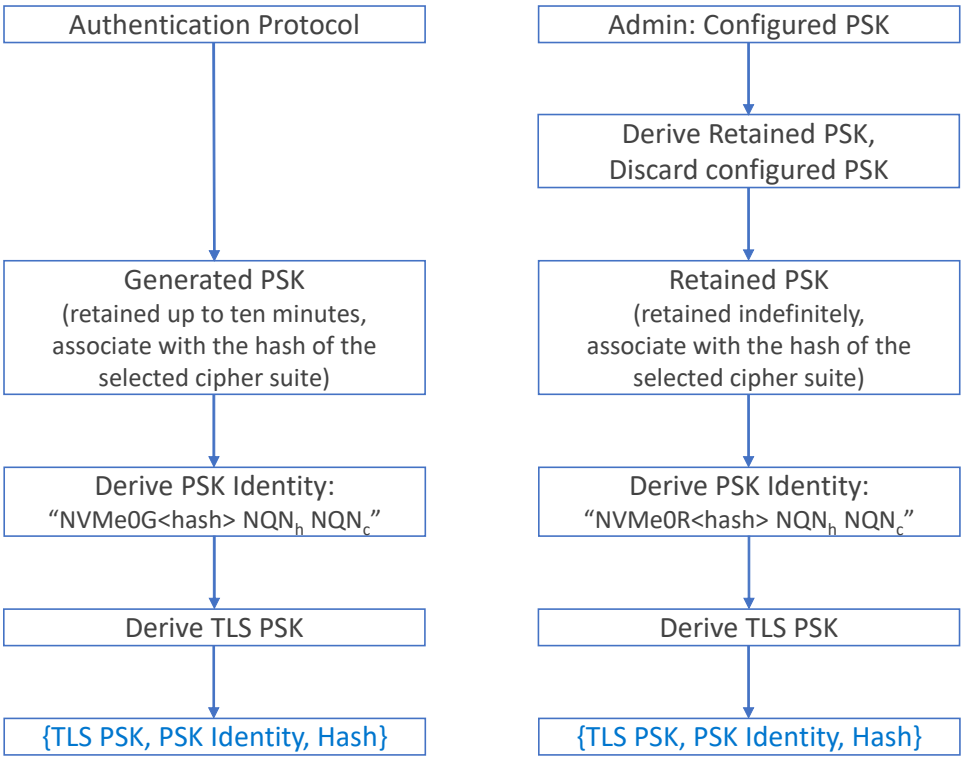

Figure TBD+1: {TLS PSK, TLS Identity, Hash} tuple derivation

7.4.9.4 PSK Reuse

A retained PSK is used between a host and an NVM subsystem to set up TLS secure channels for the Admin Queue and all I/O Queues of each controller associated with that host.

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.

Note to the editor: 6.5.9 is a section of TP 8006.

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.

7.4.9.5 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 7.4.9.3), 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 (4 bytes binary).

As an example, the 32-byte configured PSK:
`5512dbb6_737d0106_f65975b7_73dfb011_ffc344bc_f442e2dd_6d8bc487_0b5d5b03h`
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.

7.4.9.6 TLS Implementations and Use Requirements

~~The following requirements apply to use of TLS 1.2 with NVMe/TCP:~~

~~1. TLS compression shall not be used, as it is not secure (refer to section 3.3 of RFC 7525). This NVMe/TCP prohibition of TLS compression is stronger than the requirements in RFC 7525 because NVMe/TCP is a new protocol;~~

~~2. If TLS session resumption is supported, the implementation of session resumption shall comply with the requirements in section 3.4 of RFC 7525; and~~

~~3. If TLS renegotiation is supported, the renegotiation_info extension shall be implemented and used for all renegotiations as described in RFC 5746 (refer to section 3.5 of RFC 7525).~~

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;
- Each individual DH group; and
- Each individual ECDH group.

~~7.4.9.3~~7.4.9.7 Transport Service Identifier