TCG

TCG Storage Core Spec Addendum: Secure Messaging

Specification Version 1.00
Revision 1.00

August 5, 2015

Contact: admin@trustedcomputinggroup.org

PUBLISHED

TABLE OF CONTENTS

Tables

1 Introduction

1.1 Document Purpose

The Storage Workgroup specifications provide a comprehensive architecture for putting Storage Devices under policy control as determined by the trusted platform host, the capabilities of the Storage Device to conform with the policies of the trusted platform, and the lifecycle state of the Storage Device as a Trusted Peripheral.

1.2 Scope and Intended Audience

This specification defines Secure Messaging for the TCG Storage Architecture Core Specification by mapping Transport Layer Security (TLS) v1.2 onto the TCG Storage communication protocol. This specification contains the following elements that can be used in conjunction with any SSC:

- Mapping/encoding of TLS v1.2 onto the TCG Storage communication protocol
- Table definitions for holding credentials
- Other aspects that would apply to any/all SSCs incorporating TLS v1.2

Any Storage Device that claims TCG Storage Secure Messaging compatibility SHALL conform to this specification.

The intended audience for this specification is both trusted Storage Device manufacturers and developers that want to use these Storage Devices in their systems.

1.3 Key Words

Key words are used to signify SSC requirements.

The Key Words "**SHALL**", "**SHALL NOT**", "**SHOULD**," and "**MAY**" are used in this document. These words are a subset of the RFC 2119 key words used by TCG, and have been chosen since they map to key words used in T10/T13 specifications. These key words are to be interpreted as described in [1].

In addition to the above key words, the following are also used in this document to describe the requirements of particular features, including tables, methods, and usages thereof.

- **Mandatory (M):** When a feature is Mandatory, the feature SHALL be implemented. A Compliance test SHALL validate that the feature is operational.
- **Optional (O):** When a feature is Optional, the feature MAY be implemented. If implemented, a Compliance test SHALL validate that the feature is operational.
- **Excluded (X):** When a feature is Excluded, the feature SHALL NOT be implemented. A Compliance test SHALL validate that the feature is <u>not</u> operational.
- **Not Required (N)** When a feature is Not Required, the feature MAY be implemented. No Compliance test is required.

1.4 Document References

[1]. IETF RFC 2119, 1997, "Key words for use in RFCs to Indicate Requirement Levels"

[2]. Trusted Computing Group (TCG), "TCG Storage Architecture Core Specification", Version 2.01

[3]. Trusted Computing Group (TCG), "TCG Storage Storage Interface Interactions Specification", Version 1.04

[4]. Trusted Computing Group (TCG), "TCG Storage Security Subsystem Class: Opal", Versions 1.00, 2.00, 2.01, or compatible

[5]. IETF RFC 5246, 2008, "The Transport Layer Security (TLS) Protocol", Version 1.2

[6]. IETF RFC 5487, 2009, “Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode”

[7]. IETF RFC 6655, 2012, “AES-CCM Cipher Suites for Transport Layer Security (TLS)”

[8]. IETF RFC 4279, 2005, “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)”

[9]. IETF RFC 5489, 2009, “ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)”

[10]. Internet Assigned Numbers Authority (IANA), “TLS Cipher Suite Registry”
(http://www.iana.org/assignments/tls-parameters/tls-parameters.xml#tls-parameters-3)

1.5 Document Precedence

In the event of conflicting information in this specification and other documents, the precedence for requirements is:

1. This specification
2. IETF RFC 5487 [6]
3. IETF RFC 6655 [7]
4. IETF RFC 5246 [5]
5. Storage Interface Interactions Specification [3]
6. TCG Storage Architecture Core Specification [2]

1.6 Terminology

This section provides special definitions that are not defined in the Core Specification.

Table 1 Secure Messaging Terminology

Term	Definition
TLS	Transport Layer Security
PSK	Pre-Shared Key
GCM	Galois-Counter Mode
CCM	Counter with CBC-MAC

2 Secure Messaging Overview

Begin Informative Content

TCG Storage Secure Messaging as defined in this document maps version 1.2 of the Transport Layer Security (TLS) protocol onto the TCG Storage communication protocol to enable end-to-end secure messaging from an endpoint on the host (either local or remote) to an SP in the TPer. The objective is to define a limited set of requirements necessary for support by the storage device in order to meet the envisioned use cases. For example, Secure Messaging does not require support for certain TLS v1.2 features, such as compression (prohibited), session renegotiation, and session resumption. This specification addresses only PSK-based cipher suites.

Use Cases:

1. From a central administration console, the IT administrator updates the security configuration of one or more TCG-based storage devices in the organization's client computers. The communication sessions are established using Secure Messaging, ensuring end-to-end confidentiality and integrity of the communications between the administration console and each storage device, thereby protecting sensitive information (such as Administrator credentials) from compromise.
2. The local authentication application on a client computer collects a user's authentication credentials in order to authenticate to the local TCG-based storage device and perform some action on the security subsystem. The communication session is established using Secure Messaging, protecting the authentication credentials from attacks such as sniffing the storage interface (e.g. SATA cable).

Security Considerations:

New attacks against the TLS protocol are discovered from time-to-time. Implementers are recommended to seek separate guidance on various TLS attacks and mitigation options.

End Informative Content

3 New Functionality

This section defines new functionality (not contained in [2] or [3]) required to support Secure Messaging.

The methods and protocol mapping defined in this specification supersede the secure session start up methods (`StartTrustedSession`, `SyncTrustedSession`) and “Secure Messaging Packet Format” as specified in [2].

3.1 New Methods

This section defines the new methods that are required to support Secure Messaging. The new methods introduced are Session Manager Methods. The new methods are analogous to the existing StartSession/SyncSession methods, but allow for the startup of secure sessions using the TLS v1.2 protocol.

3.1.1 StartTLS

The `StartTLS` method is a Session Manager Method used to initiate a TLS session from the host (TLS client) to the TPer (TLS server). This method is invoked in place of `StartSession` when starting up a session with secure messaging.

```
SMUID.StartTLS[
    HostSessionID : uinteger,
    SPID : uidref {SPObjectUID},
    Write : boolean,
    SessionTimeout = uinteger,
    TransTimeout = uinteger,
    InitialCredit = uinteger ]
=>
SyncTLS[ see SyncTLS definition in 3.1.2]
```

Method UID: `00 00 00 00 00 00 FF 12`

3.1.1.1 HostSessionID

The definition of the **HostSessionID** parameter is the same as for the `StartSession` method defined in [2].

3.1.1.2 SPID

The definition of the **SPID** parameter is the same as for the `StartSession` method defined in [2].

3.1.1.3 Write

The definition of the **Write** parameter is the same as for the `StartSession` method defined in [2].

3.1.1.4 SessionTimeout

The definition of the **SessionTimeout** parameter is the same as for the `StartSession` method defined in [2].

3.1.1.5 TransTimeout

The definition of the **TransTimeout** parameter is the same as for the `StartSession` method defined in [2].

3.1.1.6 InitialCredit

The definition of the **InitialCredit** parameter is the same as for the `StartSession` method defined in [2].

3.1.2 SyncTLS

The `SyncTLS` method is a Session Manager Method, returned by the TPer (TLS server) in response to the invocation of the `StartTLS` method by the host (TLS client).

```
SMUID.StartTLS[see StartTLS definition in 3.1.1]
=>
SyncTLS[
    HostSessionID : uinteger,
    SPSessionID : uinteger,
    TransTimeout = uinteger,
    InitialCredit = uinteger
]
```

Method UID: `00 00 00 00 00 00 FF 13`

3.1.2.1 HostSessionID

The definition of the **HostSessionID** parameter is the same as for the `SyncSession` method defined in [2].

3.1.2.2 SPSessionID

The definition of the **SPSessionID** parameter is the same as for the `SyncSession` method defined in [2].

3.1.2.3 TransTimeout

The definition of the **TransTimeout** parameter is the same as for the `SyncSession` method defined in [2].

3.1.2.4 InitialCredit

The definition of the **InitialCredit** parameter is the same as for the `SyncSession` method defined in [2].

3.2 New Tables

This section contains new tables that are used in support of Secure Messaging.

3.2.1 Credential Table Group – C_TLS_PSK (Object Table)

Column Number	Column	IsUnique	Column Type
0x00	UID		uid
0x01	Name	Yes	name
0x02	CommonName	Yes	name
0x03	Enabled		boolean
0x04	PSK		psk
0x05	CipherSuite		bytes_2

UID of Table Descriptor Object	UID of Table	Table Name	Template
00 00 00 01 00 00 00 1E	00 00 00 1E 00 00 00 00	C_TLS_PSK	Base

3.2.1.1 UID

This is the unique identifier of this row in the `C_TLS_PSK` table.

This column SHALL NOT be modifiable by the host.

3.2.1.2 Name

This is the name of the `C_TLS_PSK` object.

For `C_TLS_PSK` objects that exist at issuance, this column SHALL NOT be modifiable by the host.

3.2.1.3 CommonName

This is a name that MAY be shared by multiple `C_TLS_PSK` objects.

For `C_TLS_PSK` objects that exist at issuance, this column SHALL NOT be modifiable by the host.

3.2.1.4 Enabled

This column identifies whether the PSK credential is enabled, thus identifying if the `C_TLS_PSK` object can be used for TLS session startup. When this value is True, this credential is enabled.

See 5.3.1.2 for details on device behavior when the host attempts to open a session using a PSK whose Enabled column is False.

3.2.1.5 PSK

This column stores the pre-shared key (PSK) associated with this `C_TLS_PSK` object.

3.2.1.6 CipherSuite

This column identifies the TLS cipher suite that SHALL be used with this PSK for TLS session startup. This column contains a two-byte encoding of the cipher suite, as specified in [10]. See 5.3.1.2 for details on device behavior when the host attempts to open a session using a cipher suite that is different from the one contained in this column.

3.3 New Types

This section contains new tables that are used in support of Secure Messaging.

3.3.1 bytes_2

This is a bytes type with a size requirement of 2.

Table 2 bytes_2

UID	Name	Format
00 00 00 05 00 00 02 04	bytes_2	Simple_Type, bytes, 2

3.3.2 psk

This is a max bytes type that provides a SSC-specific size. SSCs (or SSC feature sets) MAY impose a minimum requirement on the size that must be supported (e.g. SSC or feature set could require max_bytes_64)

Table 3 psk

UID	Name	Format
00 00 00 05 00 00 02 0F	psk	Simple_Type, max_bytes, *

4 Secure Messaging Requirements

This section defines the Mandatory (M) and Optional (O) requirements for Secure Messaging.

4.1 Level 0 Discovery

A SD that contains the Secure Messaging feature set SHALL return the Secure Messaging Feature Descriptor as described in 4.1.1.

Begin Informative Content

Level 0 Discovery data is not cryptographically protected, but can be used to let the host determine aspects of the device’s TLS support that would otherwise need to be done through trial-and-error session startups, such as the cipher suites supported by the device.

The information contained in Level 0 Discovery should not in any way be relied upon as a replacement for the handshake protocol.

End Informative Content

4.1.1 Secure Messaging Feature Descriptor (Feature Code = 0x0004)

This feature descriptor SHALL be returned when the SD supports Secure Messaging. The contents of the feature descriptor are defined in Table 4.

Begin Informative Content

Note that this feature descriptor is only for PSK-based cipher suites; any support for RSA/ECC cipher suites will have their own Level 0 feature descriptor in a separate document or a future version of this document.

End Informative Content

Table 4 Level 0 Discovery – Secure Messaging Feature Descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB) Feature Code							
1	Feature Code							(LSB)
2	Version				Reserved			
3	Length							
4	Activated	TLS Features						
5-7	Reserved							
8-9	Number of SPs (m)							
10-17	SP 1							
18-25	SP 2							
…	…							
(8m+2) to (8m+9)	SP m							
(8m+10) to (8m+11)	Number of Cipher Suites Supported (n)							
(8m+12) to (8m+15)	Cipher Suite 1							
(8m+16) to (8m+19)	Cipher Suite 2							
…	…							
(8m+4n+8) to (8m+4n+11)	Cipher Suite n							

4.1.1.1 Activated

This field indicates whether Secure Messaging is currently activated:

- 0 –Secure Messaging is currently not activated
- 1 –Secure Messaging is currently activated

Storage Devices that do not support field activation of features SHALL report a value of 1 in the Activated field.

Begin Informative Content

The Activated field is intended to report whether a field-activatable feature has been activated. Field activation of features is outside the scope of this specification. Storage devices that do not support field activation of features but do support Secure Messaging will report a value of 1 in the Activated field.

End Informative Content

4.1.1.2 TLS Features

This field indicates whether the Storage Device supports certain TLS features:

- **Bits 6-5 – Reserved**

- **Bit 4 – CertificateRequest:**
    - 0 – Storage Device does not support client authentication via the CertificateRequest message
    - 1 – Storage Device supports client authentication via the CertificateRequest message
- **Bit 3 – Server Certificate:**
    - 0 – Storage Device does not contain a Server Certificate (e.g. only PSK-based cipher suites are supported)
    - 1 – Storage Device contains one or more Server Certificates
- **Bit 2 – Renegotiation:**
    - 0 – Storage Device does not support Renegotiation
        - In response to a ClientHello message sent by the client to renegotiate the security parameters of an existing session, the server is only required to be able to respond with a no_renegotiation alert.
    - 1 – Storage Device supports Renegotiation
- **Bit 1 – Compression:**
    - 0 – Storage Device does not support Compression
        - The storage device SHALL NOT support Compression.
- **Bit 0 – Session Resumption:**
    - 0 – Storage Device does not support Session Resumption
        - The session_id field in the ServerHello message may contain a value different from the value in the ClientHello message's session_id field, indicating that the client's request to resume a previous session has been rejected. Additionally, the server may return an empty session_id to indicate that the session will not be cached and therefore cannot be resumed.
    - 1 – Storage Device supports Session Resumption

4.1.1.3 Number of SPs

This value indicates the number of SPs in the TPer that support Secure Messaging.

4.1.1.4 SP 1 … SP m

These fields contain a list of SPs in the TPer that support Secure Messaging. Each entry in the list contains the SP's UID from the SP table of the Admin SP.

4.1.1.5 Number of Cipher Suites Supported

This value indicates the number of Cipher Suites supported by the TPer.

4.1.1.6 Cipher Suite 1 … Cipher Suite n

These fields contain the cipher suites supported by the device. Each entry in the list is a four-byte value containing the encoding of the cipher suite, as specified in [10], with 2 zero bytes pre-pended.

The Storage Device SHOULD list its cipher suites in order of device preference, highest to lowest.

4.1.1.7 Level 0 requirements for Secure Messaging

- **Feature Code**: 0x0004
- **Version**: 0x1 or any version that supports the defined features in this specification
- **Length**: Variable – depends on the number of supported cipher suites

5 TLS Requirements

This section describes the requirements for supporting the TLS v1.2 protocol [5] over the TCG Storage communication protocol.

5.1 TLS Server

An SD supporting Secure Messaging SHALL be the TLS Server.

5.2 TLS Session Startup

TLS session startup is accomplished by utilizing the two Session Manager methods described in section 3.1:

- `StartTLS`
- `SyncTLS`

The `StartTLS` and `SyncTLS` methods are used to establish the target SP of the session, the Session number (comprised of the TSN and HSN as described in [2]) that will be used in the Session field of all packets for the associated session, session timeout parameters, etc. The TCG Session number has no relationship to the TLS session identifier.

Once the TCG session has been established, all TLS-related traffic occurs within TCG Packets exchanged between the host and Storage Device using the agreed-upon Session number, as described in 5.3.

5.3 TLS Record Protocol

After successful session startup using the `StartTLS` and `SyncTLS` methods, all TLS traffic occurs within Packets exchanged between the host and the Storage Device using the Session number agreed upon in the `StartTLS` and `SyncTLS` methods.

The payloads of the Packets SHALL be TLS records as shown in Figure 1.

All sub-protocols of the TLS record protocol are communicated in the Packet payloads: the handshake protocol, the alert protocol, the change cipher spec protocol, and the application data protocol (see [5] for details on the TLS record protocol and the four sub-protocols).

In order to minimize the number of host-to-device round trips during the TLS handshake protocol, and to support TLS application data that is longer than the maximum TLSPlaintext fragment length (as defined in [5]), the SD SHALL be capable of:

1. Accepting multiple TLS handshake protocol records within Packets received from the host.
2. Creating multiple TLS handshake protocol records in its response Packets to the host.

The SD SHALL be capable of sending and receiving multiple TLS application records in a single packet, with the following conditions:

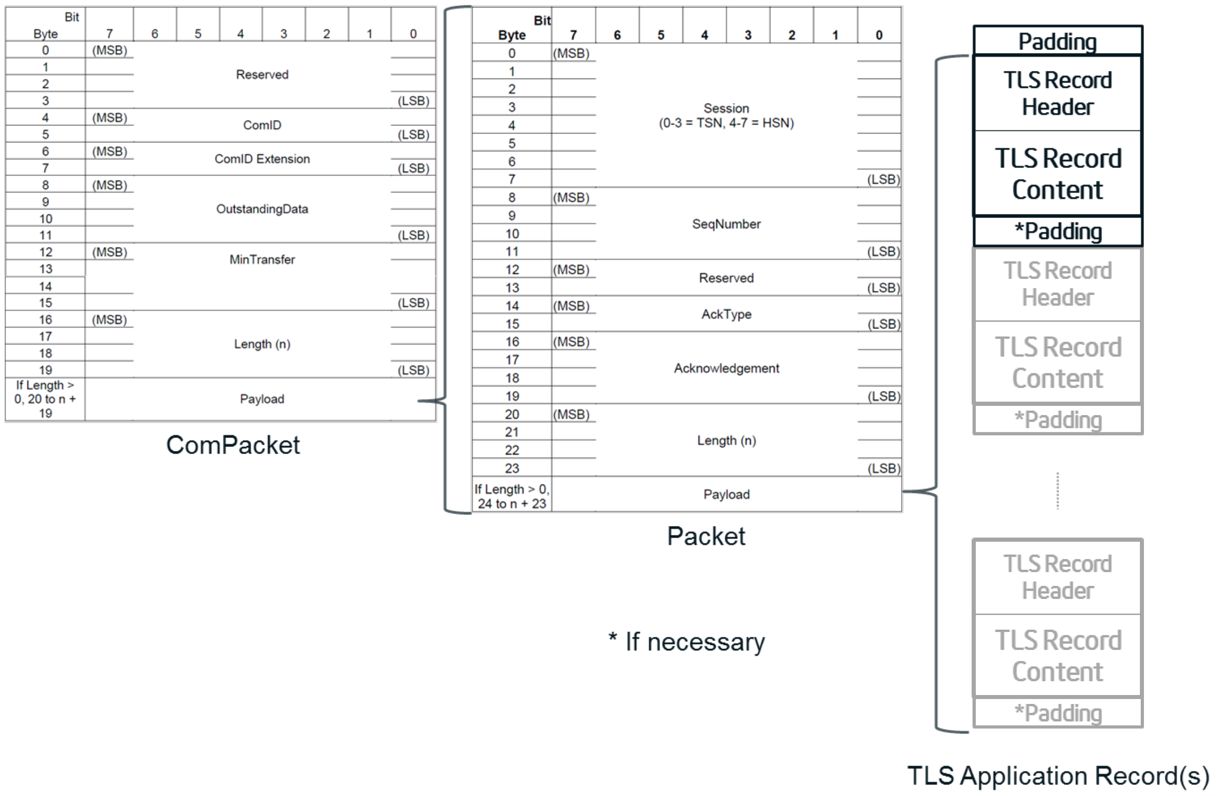
1. Multiple TLS application records SHALL be used if
   a. The lesser of the sender's maximum supported Packet length and the receiver's MaxPacketSize property is greater than the maximum TLSCiphertext length (plus TLS header plus pad), and
   b. The communicator is attempting to send more data than the maximum TLSPlaintext fragment length.
2. If the SD is incapable of sending packets larger than the maximum TLSCiphertext length (plus TLS header plus pad), the SD SHALL send its response as a single TLS application record.

When encoding multiple TLS application records into Packet payloads:

1. The SD SHALL utilize maximally-sized TLS records except for the final record within the Packet. The final TLS record within the Packet MAY be smaller than the maximum size.

2. The host SHOULD utilize maximally-sized TLS records except for the final record within the Packet, which MAY be smaller than the maximum size.

The bytes of TLS records SHALL be ordered within the Packet as specified in [5] (network byte order).



Byte	Bit 7	6	5	4	3	2	1	0
0	(MSB)	Reserved						
1								
2								
3								(LSB)
4	(MSB)	ComID						
5								(LSB)
6	(MSB)	ComID Extension						
7								(LSB)
8	(MSB)	OutstandingData						
9								
10								
11								(LSB)
12	(MSB)	MinTransfer						
13								
14								
15								(LSB)
16	(MSB)	Length (n)						
17								
18								
19								(LSB)
If Length > 0, 20 to n + 19	Payload							

Byte	Bit 7	6	5	4	3	2	1	0
0	(MSB)	Session (0-3 = TSN, 4-7 = HSN)						
1								
2								
3								
4								
5								
6								
7								(LSB)
8	(MSB)	SeqNumber						
9								
10								
11								(LSB)
12	(MSB)	Reserved						
13								(LSB)
14	(MSB)	AckType						
15								(LSB)
16	(MSB)	Acknowledgement						
17								
18								
19								(LSB)
20	(MSB)	Length (n)						
21								
22								
23								(LSB)
If Length > 0, 24 to n + 23	Payload							

Figure 1 TLS Records within TCG Packets

5.3.1 TLS Handshake Protocol

This section details requirements for the TLS handshake protocol as it relates to this specification. For details on the TLS handshake protocol, refer to [5].

5.3.1.1 CipherSuite Requirements

An SD supporting Secure Messaging SHALL support, at a minimum, the following cipher suite (see [6]):

- TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 {0x00,0xAA}

The following cipher suites are Optional (O) (see [6], [7], and [9]):

- TLS_PSK_WITH_AES_128_GCM_SHA256 {0x00,0xA8}
- TLS_PSK_WITH_AES_256_GCM_SHA384 {0x00,0xA9}
- TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 {0x00,0xAB}
- TLS_PSK_WITH_AES_128_CBC_SHA256 {0x00,0xAE}
- TLS_PSK_WITH_AES_256_CBC_SHA384 {0x00,0xAF}
- TLS_PSK_WITH_NULL_SHA256 {0x00,0xB0}
- TLS_PSK_WITH_NULL_SHA384 {0x00,0xB1}

- TLS_DHE_PSK_WITH_AES_128_CBC_SHA256 {0x00,0xB2}
- TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 {0x00,0xB3}
- TLS_DHE_PSK_WITH_NULL_SHA256 {0x00,0xB4}
- TLS_DHE_PSK_WITH_NULL_SHA384 {0x00,0xB5}
- TLS_PSK_WITH_AES_128_CCM {0xC0,0xA4}
- TLS_PSK_WITH_AES_256_CCM {0xC0,0xA5)
- TLS_DHE_PSK_WITH_AES_128_CCM {0xC0,0xA6}
- TLS_DHE_PSK_WITH_AES_256_CCM {0xC0,0xA7}
- TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256 {0xC0, 0x37}
- TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 {0xC0, 0x38}
- TLS_ECDHE_PSK_WITH_NULL_SHA256 {0xC0, 0x3A}
- TLS_ECDHE_PSK_WITH_NULL_SHA384 {0xC0 ,0x3B}

Begin Informative Content

The "DHE" cipher suites are preferred because they provide perfect forward secrecy. It is recommended to utilize the "DHE" cipher suites when the TLS session will carry valuable long-term secrets, such as Admin passwords.

End Informative Content

5.3.1.2 PSK Identities

The PSKs to be used for the TLS handshake protocol SHALL be contained in the target SP's C_TLS_PSK table as specified in section 3.2.1.

During the TLS handshake protocol, the host (TLS client) selects the PSK to use by encoding the UID of the desired C_TLS_PSK object in the psk_identity field of the ClientKeyExchange message. The psk_identity is encoded such that each hex digit of the hexadecimal UID is encoded in its UTF-8 representation. Hex letters (A-F) are encoded in uppercase (A, B, C, D, E, F). When the UID is encoded in this fashion, it will always result in a 16 byte PSK identity value.

For any of the following conditions, the Storage Device SHALL proceed with the TLS handshake as if the PSK existed, was enabled, and had the correct cipher suite selected by the host, but the key was incorrect (that is, respond with a "decrypt_error" alert):

- The UID value as encoded in the psk_identity does not exist in the C_TLS_PSK table on the device
- The `Enabled` column of the PSK object that corresponds to the UID value as encoded in the psk_identity is set to False
- The host selected a cipher suite other than the one contained in the identified PSK object's `CipherSuite` column

Begin Informative Content

The TLS PSK standard allows the server (device) to potentially return an "unknown_psk_identity" alert message if the psk_identity is not recognized. This specification requires the device to return a "decrypt_error" alert, as noted above. This specification also requires the device to return "decrypt_error" when the PSK object does not exist on the device; the PSK object is disabled on the device; or a mismatch is detected between the host selected cipher suite and the one associated with the PSK object as stored on the device. The device is only able to check these error conditions after it has received the client's (host) TLS handshake protocol Finished message.

End Informative Content

5.3.1.3 Compression

In its ServerHello message, the SD SHALL indicate selection of the “null” compression method.

5.3.2 TLS Application Protocol

After the TLS handshake protocol completes successfully, communication between the host and the target SP commences as normal, with the exception that the Packet payloads are TLS application records as shown in Figure 2. The TLS application records contain encrypted/MACed TCG Subpackets. Per [5], “Client message boundaries are not preserved in the record layer (i.e., multiple client messages of the same ContentType MAY be coalesced into a single TLSPlaintext record, or a single message MAY be fragmented across several records).” As such, a single TLS application record MAY contain multiple TCG Subpackets, or a single TCG Subpacket may be fragmented across several TLS application records.

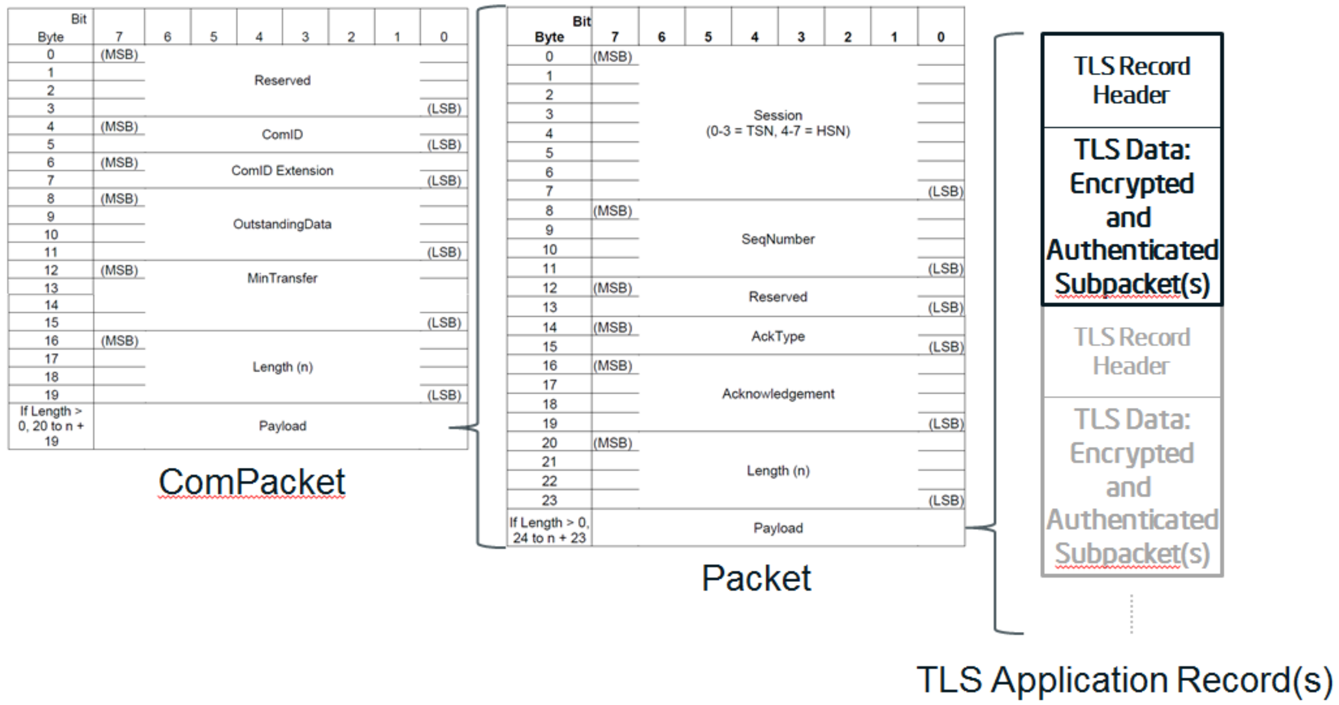

Bit
Byte
7 6 5 4 3 2 1 0
0–3 (MSB) Reserved (LSB)
4–5 (MSB) ComID (LSB)
6–7 (MSB) ComID Extension (LSB)
8–11 (MSB) OutstandingData (LSB)
12–15 (MSB) MinTransfer (LSB)
16–19 (MSB) Length (n) (LSB)
If Length > 0, 20 to n + 19 Payload
ComPacket
0–7 (MSB) Session (0-3 = TSN, 4-7 = HSN) (LSB)
8–11 (MSB) SeqNumber (LSB)
12–13 (MSB) Reserved (LSB)
14–15 (MSB) AckType (LSB)
16–19 (MSB) Acknowledgement (LSB)
20–23 (MSB) Length (n) (LSB)
If Length > 0, 24 to n + 23 Payload
Packet
TLS Record Header
TLS Data: Encrypted and Authenticated Subpacket(s)
TLS Record Header
TLS Data: Encrypted and Authenticated Subpacket(s)
TLS Application Record(s)

Figure 2 TLS Application Protocol Records

5.3.3 Interactions between TLS and the TCG Storage Communications Protocol

This section specifies the interactions between the TLS protocol and the TCG Storage Communications Protocol.

5.3.3.1 Padding

In order to align the payloads of the TLS records to dword offsets within the Packet, there SHALL be 3 bytes of Pad (0x00) preceding the first TLS record in the Packet. Each TLS record in the Packet, including the final TLS record in the Packet, SHALL be followed by 0 to 3 bytes of Pad (0x00), determined by the following formula based on the prior TLS record's length field: ((3 - length) modulo 4). In addition to the 0-3 bytes of pad added after the final TLS record in the Packet, one extra byte of Pad (0x00) SHALL be added after the final TLS record in the Packet (for 0-3 plus 1 bytes of pad total), to ensure dword offset alignment of the next Packet (if any) in the ComPacket.

Data Subpackets within the TLS application record(s) SHALL be padded as defined in [2].

Begin Informative Content

Note that the standard C library does not handle the modulo operation with negative numbers properly.

End Informative Content

5.3.3.2 PSK Length

Begin Informative Text

Per [8], PSK sizes can be any size, up to 2^16-1 bytes, and are not required to be the size of the key for the encryption + MAC algorithms.

End Informative Text

SDs SHALL support arbitrary PSKs up to 64 octets in length.

The host SHOULD use a PSK length that is sized appropriately for the strength of the cipher suite being used.

5.3.3.3 Ending Sessions

After the host sends an End of Session (EOS) token to close the TCG Session, the host SHOULD send a TLS close_notify alert to close the TLS session. The host SHOULD NOT send a TLS close_notify alert until after it has sent the (EOS) token on the TCG session.

If the SD receives a TLS close_notify alert prior to receiving EOS in the TCG session, the SD SHALL abort the TCG Session, and SHALL perform all side effects of aborting a session as described in [2].

After the SD receives End of Session in the TCG session, any additional data received in the encapsulating TLS session SHALL be ignored and discarded.

After the SD sends its EOS in response to an EOS from the host, the SD SHOULD immediately send a close_notify alert, even if the host has not yet sent a close_notify. Per the TLS specification, the initiator of close_notify is not required to wait for the responding close_notify before closing both the read side and the write side of its connection (see “Closure Alerts” in [5]).

After TCG session close, the TCG session number within the Packet SHALL remain valid until the host retrieves the TLS close_notify alert. The following will also invalidate the device’s TLS session identifier:

1. A TPer Programmatic Reset (see [4])
2. A ComID reset using STACK_RESET

5.3.3.4 Session Aborts

When the SD aborts the TCG session due to some error in the TCG Storage Communication Protocol (e.g. invalid tokenization of the message stream), and not related to a TLS protocol failure, the SD SHALL close the TLS session and indicate this to the host by sending a TLS close_notify alert.

In scenarios where the SD aborts the TCG session and returns the CloseSession method, the SD SHALL first return a TLS close_notify alert on the current TCG session, using the appropriate session number, before preparing CloseSession.

5.3.3.5 TLS Fatal Alerts

When the SD receives a TLS Fatal Alert from the host, the SD SHALL immediately abort the TCG session (see [2]). The SD SHALL NOT prepare any response, including a CloseSession method. The ComID SHALL transition to the Awaiting IF-SEND state.

When the SD detects a TLS Fatal Alert condition, the SD SHALL generate an appropriate TLS message; and abort the TCG session (see [2]). The SD SHALL NOT prepare a CloseSession method.

5.3.3.6 Timeouts

All timeouts associated with the session SHALL be enforced at the TLS record layer of the protocol stack. Examples:

- If the TLS handshake protocol does not complete within the established SessionTimeout, the TLS session SHALL be aborted (even though no communications with the actual SP have started).
- If the host has ended the TCG Session by sending the EOS token, but has not sent a TLS close_notify alert, then in the absence of any alert that would cause the TLS session to close/abort (such as a close_notify alert or fatal alert), the session SHALL remain open until the SessionTimeout value has been reached. Note that the SD SHOULD send a close_notify alert after sending its EOS token, even if the host has not sent its close_notify alert, thus avoiding the need to wait for SessionTimeout to expire (see 5.3.3.1).

Sessions aborted due to timeouts SHALL be handled as described in 5.3.3.4.