TRUSTED® COMPUTING GROUP

SPECIFICATION

TPM Communication over SPDM Secure Session

Version 1.0

December 12, 2025

Contact: admin@trustedcomputinggroup.org

Published

© TCG 2025

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
V1.0 Rev 0.36	February 13, 2025	First version for IP/Technical and Public Review

V1.0 Rev 0.42	August 22, 2025	Second version for IP/Technical and Public Review
		
		Updated/added sections:
		• 5.5.1.2. GET_CAPABILITIES
		  ○ Require PUB_KEY_ID_CAP to be SET (in the request and response)
		• 5.5.1.4.2 Opaque Data
		  ○ OpaqueData is encoded in the “general opaque data format” as defined in DSP0274, not the “Secured Messages opaque data format” as defined in DSP0277
		• 5.5.1.7 ERROR
		  ○ Error for messages that are extended into transcript hash terminate the connection
		  ○ TPM is not required to support retry of request messages
		  ○ Added the error codes Vd_TpmVendorDefined and Vd_LocalityMismatch
		• 5.5.2.6 TPM_CMD_L*
		  ○ Added (optional) locality-aware TPM_CMD_L* requests that indicate the intended locality of the encapsulated TPM command
		• 5.5.3.2.1.1 Failure Mode
		  ○ Clarified condition when TPM may enter Failure mode
		• 5.5.3.2.2 Field Upgrade
		  ○ Clarified that RC_BAD_TAG is returned if TPM does not accept SPDM messages during Field Upgrade
		• 5.7.2.2 TPM Command Cancel and Abort
		  ○ Specified how Command Cancel and Abort apply to SPDM messages
		
		Updated figures:
		• Minor correction in figure 6 Detailed SPDM message flow
		• Updated figure 16 Startup sequence to clarify Startup time

V1.0 Rev 0.44	November 17, 2025	First published version Added (informative) section: • 5.5.1.2.3 DataTransferSize ○ Recommend Requester to set DataTransferSize to at least the maximum TPM response size (plus overhead)

 © TCG 2025

Acknowledgements

The writing of a specification, particularly a security specification, takes many hours for both development and review. The TCG would like to acknowledge the contribution of those individuals (listed below) and the companies who allowed them to volunteer their time to the development of this specification. Special thanks are due to Shiva Dasari, who served as Chair of the Server Working Group, and Ga-Wai Chin, who was the editor during the development of this specification.

Organization	Contributors
ARM Ltd.	Stuart Yoder
Cisco Systems	Doug Ambrisko
Dell, Inc.	Amy Nelson, Travis Gilbert, Eugene Cho
Google Inc.	Jeff Andersen, Chris Fenner, Jordan Hand
Hewlett Packard Enterprise	Shiva Dasari, Tom Laffey, Theo Koulouris
Huawei Technologies Co., Ltd.	Thomas Deleuran
IBM	Chris Engel
Infineon Technologies	Ga-Wai Chin, Andreas Fuchs, Sven Schuch, Ricardo Schwarzmeier, Daniel Flaschka
Intel Corporation	Jiewen Yao, Xiaoyu Ruan, Thomas Bowen
Lenovo (United States) INC	Michael Demeter
Meta Platforms Inc.	Jonathan McDowell
Microsoft	Ahmad Abdullateef, Vishal Soni
NVIDIA Corporation	Ludovic Jacquin, Joe Pennisi, Steven Bellock
STMicroelectronics	Fabien Arrivé, Yves Magnaud, Laurent Charpentier
Phoenix Technologies	Dick Wilkins, Tim Hoppen

 © TCG 2025

CONTENTS

DISCLAIMERS, NOTICES, AND LICENSE TERMS 1
CHANGE HISTORY 2
Acknowledgements 3
TABLES 7
FIGURES 8
1 SCOPE 9
1.1 Introduction 9
1.2 SPDM Functionality Scope 9
1.3 Requester Scope 9
1.4 Responder (TPM) Scope 9
1.5 Platform Scope 9
1.6 Key Words 9
1.7 Statement Type 9
2 REFERENCES 11
3 TERMS AND DEFINITIONS 12
3.1 Acronyms 12
3.2 Terminology 12
4 INTRODUCTION 13
4.1 Motivation 13
4.2 Advantages 13
4.2.1 Non-disruptive to existing host applications 13
4.2.2 Attestable channel 13
4.2.3 Full encryption 13
4.3 Remotely-Attested Channel 13
5 TPM SPDM SECURE TRANSPORT PROTOCOL 15
5.1 SPDM Specifications 15
5.2 TPM Supported SPDM Functionality 15
5.3 SPDM Secure Session Layer 15
5.3.1 Architecture of a TPM Implementation with an SPDM Layer 16
5.4 Backward Compatibility 17
5.4.1 TPM Command Acceptance 17
5.4.2 FIFO/ CRB Interface and Locality 17
5.5 SPDM Messages 18
5.5.1 SPDM Requests and Responses 18
5.5.1.1 GET_VERSION 19

 © TCG 2025

5.5.1.2 GET_CAPABILITIES 19
5.5.1.2.1 MUT_AUTH_CAP 20
5.5.1.2.2 CTExponent 21
5.5.1.2.3 DataTransferSize 22
5.5.1.3 NEGOTIATE_ALGORITHMS 22
5.5.1.3.1 SPDM Algorithms 22
5.5.1.3.1.1 SPDM Specification Supported Algorithms 23
5.5.1.3.1.2 Algorithm Selection 23
5.5.1.3.2 Opaque Data Format 24
5.5.1.4 KEY_EXCHANGE 24
5.5.1.4.1 Slot ID 24
5.5.1.4.2 Opaque Data 24
5.5.1.4.2.1 Secured Message Version 25
5.5.1.4.3 Mutual Signature Verification 25
5.5.1.5 FINISH 25
5.5.1.5.1 Slot ID 25
5.5.1.5.2 Mutual Signature Verification 25
5.5.1.5.3 ResponderVerifyData 26
5.5.1.6 END_SESSION 26
5.5.1.7 ERROR 26
5.5.1.7.1 Vendor-defined Error Format 28
5.5.2 Vendor-defined Requests and Responses 28
5.5.2.1 Transcript Hash Considerations for *_PUB_KEY 29
5.5.2.2 Vendor-defined Request and Response Payload 30
5.5.2.3 GET_PUB_KEY 30
5.5.2.4 GIVE_PUB_KEY 31
5.5.2.5 TPM_CMD 31
5.5.2.6 TPM_CMD_L* 31
5.5.3 SPDM Message Flow 32
5.5.3.1 Message Flow Requirements 32
5.5.3.1.1 SPDM Request and Response Messages Validity 35
5.5.3.1.2 TPM vs. SPDM Response 36
5.5.3.2 Special Cases 36
5.5.3.2.1 Error Cases 36
5.5.3.2.1.1 Failure Mode 36
5.5.3.2.2 Field Upgrade 39

 © TCG 2025

5.6 SPDM Payload Format 39
5.6.1 SPDM Payload Byte Ordering 41
5.6.2 SPDM Payload Tags 42
5.6.3 Connection Handle 43
5.6.4 FIPS Service Indicator 43
5.6.5 SPDM Secured Message Format 45
5.7 SPDM Connections, Sessions 46
5.7.1 SPDM Connection 47
5.7.2 SPDM Session 48
5.7.2.1 Special TPM Commands and Signals 50
5.7.2.1.1 TPM_Hash_Start/ Data / End Signals 50
5.7.2.1.2 Startup Sequence 50
5.7.2.2 TPM Command Cancel and Abort 51
5.7.2.2.1 Command Cancel 51
5.7.2.2.2 Command Abort 52
5.7.2.3 Enforcing SPDM Session Protection via TPM Policy 52
5.7.3 Terminating a Connection, Session 53
6 SPDM-IDENTITY KEYS 54
6.1 Object Public Structure 54
6.1.1 Object Attributes (TPMA_OBJECT) 55
6.1.2 Object Public Structures (TPMT_PUBLIC) 55
6.1.2.1 ECC NIST P-384 55
6.1.2.2 ECC NIST P-521 56
6.2 TPM SPDM-Identity Key 56
6.2.1 Creation, Usage and Lifetime 56
7 SPDM USE CASES 58
7.1 Use case 1: Remotely-attested channel (Server, Datacenter) 58
7.1.1 General 58
7.1.2 Detailed Requirements 58
8 TPM VENDOR-SPECIFIC SUPPORT 60

 © TCG 2025

TABLES

Table 1: Applicability of TCG specifications 16
Table 2: Mandatory SPDM commands 18
Table 3: Setting of SPDM Capabilities 20
Table 4: Mutual and Responder-only signature verification 21
Table 5: SPDM algorithm sets 22
Table 6: Secured Message data 25
Table 7: Secured Message version 25
Table 8: Error codes 27
Table 9: Vendor-defined error code values 28
Table 10: Mandatory vendor-defined SPDM commands 29
Table 11: Vendor-defined request and response code and payload 30
Table 12: SPDM request and response messages validity 35
Table 13: Error cases 36
Table 14: Failure Mode error cases 37
Table 15: SPDM message field byte ordering 42
Table 16: SPDM message tags 42
Table 17: FIPS 140-3 Service Indicator 44
Table 18: Approved status of SPDM requests 44
Table 19: Overall Approved status of TPM_CMD request 45
Table 20: Secured message fields 46
Table 21: Command/ events to terminate connection or session 53
Table 22: SPDM-identity key object attributes (TPMA_OBJECT) 55
Table 23: TPMA_OBJECT values 55
Table 24: SPDM-identity key public structure (TPMT_PUBLIC) for ECC NIST P-384 56
Table 25: SPDM-identity key public structure (TPMT_PUBLIC) for ECC NIST P-521 56
Table 26: Use case 1 – Detailed Requirements 59
Table 27: Vendor-specific support 60

 © TCG 2025

FIGURES

Figure 1: Protocol layers in a TPM implementation with an SPDM layer 16
Figure 2: TPM implementation with SPDM layer architecture 17
Figure 3: Vendor-defined ERROR 28
Figure 4: SPDM message flow with mutual signature verification 32
Figure 5: SPDM message flow with Responder-only signature verification 33
Figure 6: Detailed SPDM message flow 34
Figure 7: Error cases 36
Figure 8: Failure Mode error cases 38
Figure 9: SPDM clear response message TPM_RSP 38
Figure 10: Exchange of Failure Mode information 39
Figure 11: SPDM payload format 40
Figure 12: Vendor-defined request/ response 41
Figure 13: TPM command encapsulated in SPDM secured message 41
Figure 14: SPDM connections and sessions 49
Figure 15: Example with multiple connections 50
Figure 16: Startup sequence 51

 © TCG 2025

1 SCOPE

1.1 Introduction

This TPM Communication over SPDM Secure Session specification defines the use of the DMTF Security Protocol and Data Model (SPDM) [1] to provide secure communication between a host and its TPM.

The goal is to establish a secure channel between a host (acting as Requester) and a TPM (acting as Responder): however, the goal is not to enable the TPM to be a full, generic SPDM Responder.

This specification defines the subset of SPDM functionalities required to establish a secure channel with a TPM. SPDM secured messages are used to encapsulate TPM commands and responses to provide them with confidentiality and integrity protection.

Further, this specification defines the:

- SPDM payload format/ fields – used to transport SPDM messages to a TPM
- TPM's SPDM signing key – used by a TPM for secure channel establishment

This specification defines requirements for a TPM (to enable support for an SPDM secure channel); it does not define requirements for a host.

1.2 SPDM Functionality Scope

The following SPDM functionality is either in the scope of this specification or not:

In scope:

- Secure communication: A Requester and TPM use an SPDM session for the transmission of encrypted and integrity-protected TPM commands

Not in scope:

- Device attestation: A TPM does not use SPDM measurements to attest TPM firmware
- Certificate exchange: A TPM does not use X.509 certificates to authenticate the Requester's identity

1.3 Requester Scope

This specification allows the Requester to be any type of microcontroller that controls the physical interface to the TPM (such as a host CPU or a BMC).

1.4 Responder (TPM) Scope

The secure transport protocol described in this specification may be implemented by any type of TPM, but it is most useful for discrete TPMs. That is because the data bus can be physically accessed if the data bus between a host and a TPM is not physically encapsulated.

1.5 Platform Scope

This specification is designed for Server and Data Center platforms, but may be suitable for other platforms with similar requirements.

1.6 Key Words

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.7 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as

 © TCG 2025

such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statement.

Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

 © TCG 2025

2 REFERENCES

[1] DMTF, „Security Protocol and Data Model (SPDM) Specification Version 1.3.2,“ 21 August 2024. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.3.2.pdf.

[2] Trusted Computing Group, „Trusted Platform Module Library Specification Version 184,“ 20 March 2025. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/.

[3] Trusted Computing Group, „CPU to TPM Bus Protection Guidance – Passive Attack Mitigations Version 1.0,“ 21 April 2025. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG_CPU_TPM_Bus_Protection_Guidance_Passive_Attack_Mitigation_v1p0_PUB.pdf.

[4] Trusted Computing Group, „CPU to TPM Bus Protection Guidance – Active Attack Mitigations Version 1.0,“ 8 May 2023. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-cpu-to-tpm-bus-protection-guidance-for-active-attacks/.

[5] Open Compute Project, „Datacenter Secure Control,“ 11 March 2021. [Online]. Available: https://www.opencompute.org/documents/ocp-dc-scm-spec-rev-1-0-pdf.

[6] DMTF, „Secured Messages using SPDM Specification Version 1.2.0,“ 10 June 2024. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0277_1.2.0.pdf.

[7] Trusted Computing Group, „TCG PC Client Platform TPM Profile (PTP) Specification Version 1.06,“ 30 April 2025. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/PC-Client-Specific-Platform-TPM-Profile-for-TPM-2p0-Version-1p06_pub.pdf.

[8] Internet Engineering Task Force, „HMAC-based Extract-and-Expand Key Derivation Function (HKDF),“ May 2010. [Online]. Available: https://datatracker.ietf.org/doc/rfc5869/.

[9] NIST, „Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program,“ 26 March 2024. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf.

[10] Trusted Computing Group, „TCG FIPS 140-3 Guidance for TPM 2.0 Version 1.0,“ 28 June 2024. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-fips-140-3-guidance-for-tpm-2-0/.

 © TCG 2025

3 TERMS AND DEFINITIONS

For the purposes of this specification, the following terms and definitions apply.

3.1 Acronyms

ABBREVIATIONS	DESCRIPTION
CRB	Command Response Buffer
DC-SCM	Datacenter Secure Control Module
DICE	Device Identifier Composition Engine
DMTF	Distributed Management Task Force
OCP	Open Compute Project
SPDM	Security Protocol and Data Model
SVN	Security Version Number
VCA	Version-Capabilities-Algorithms

3.2 Terminology

TERMS	DESCRIPTION
Encapsulated TPM command	A TPM command that is sent within an SPDM session, i.e., a TPM command that is encapsulated in an SPDM secured message
Mutual signature verification	During session handshake, an SPDM Requester and Responder exchange their public signature keys and use them to verify each other's signature
Responder-only signature verification	During session handshake, an SPDM Requester receives the Responder's public signature key and uses it to verify the Responder's signature
SPDM clear message	A plaintext SPDM request or response message that is neither encrypted nor integrity-protected
SPDM connection	A communication path between Requester and Responder started with VCA (Version-Capabilities-Algorithms)
SPDM-identity key	Requester or Responder key used in the SPDM handshake for signature generation and verification
SPDM payload	The SPDM request or response message preceded by TCG-defined (header) fields
SPDM Requester	The entity that sends an SPDM request message. In this specification, a host: such as a host CPU or a BMC
SPDM Responder	The entity that sends an SPDM response message. In this specification, a TPM
SPDM secured message	A SPDM request or response message that is encrypted and integrity-protected
SPDM session	A secure session started with an SPDM KEY_EXCHANGE message
TPM SPDM secure transport protocol	The secure transport protocol defined in this specification based on SPDM messages
TPM with SPDM	A TPM implementation with an SPDM secure session layer
Unencapsulated TPM command	A TPM command that is not sent within an SPDM session, i.e., a TPM command that is not encapsulated in an SPDM secured message

 © TCG 2025

4 INTRODUCTION

4.1 Motivation

Start of informative comment

This specification defines a secure transport protocol using SPDM to establish a secure channel between a host and a TPM to protect the physical bus (SPI, I2C) from passive and active interposer attacks.

In a passive attack, an attacker can monitor the traffic between the host and TPM and may be able to snoop plaintext secrets (e.g., that are released in a TPM2_Unseal response). In an active attack, an attacker can modify or suppress TPM commands (e.g., drop PCR extensions) or inject arbitrary TPM commands.

The existing TPM functionality defined in the TPM Library specification [2] provides mechanisms to mitigate passive and certain active interposer attacks (via TPM sessions). The mitigation mechanisms are described in the following TCG reference documents:

- CPU to TPM Bus Protection Guidance – Passive Attack Mitigations [3]
- CPU to TPM Bus Protection Guidance – Active Attack Mitigations [4]

The TPM SPDM secure transport protocol defined in this specification can be implemented instead or in addition to those mitigation mechanisms to protect the communication bus from active and passive attacks. The advantages of this protocol are described in the subsequent section.

Some platforms may implement means in hardware to mitigate an interposer from accessing the bus. Access can be made more difficult via the physical layout of the motherboard (e.g. TPM soldered to motherboard or buried traces) or by operating the platform in a secure environment. However, for some platforms, these means are not viable. One use case is datacenter platforms. New designs for datacenter servers such as those standardized by the Open Compute Project (OCP) in the DC-SCM standard [5] make interfaces, including the TPM interface, easily interposable and require parts to be swappable, which may preclude a permanent binding between a TPM and host.

End of informative comment

4.2 Advantages

Start of informative comment

This section describes the advantages of the secure transport protocol over TPM sessions.

4.2.1 Non-disruptive to existing host applications

This TPM SPDM secure transport protocol can be used to secure TPM host applications that do not employ the mitigation mechanisms in [3], [4] without requiring disruptive changes to existing host applications and users.

This TPM SPDM secure transport protocol uses SPDM secure sessions, which allow TPM commands (albeit not _TPM signals) to be transparently tunneled across a remotely-attested channel to the TPM (see section 4.3). This allows for minimal disruption to existing TPM users. The secure channel establishment can be implemented by a TPM host driver and, therefore, does not require (higher-level) TPM host applications to change.

4.2.2 Attestable channel

This protocol can be used to prove to an external verifier that a given TPM object is protected by a secure channel that was established with a particular host (signing) key.

4.2.3 Full encryption

Another advantage is that SPDM sessions allow the entire TPM command and response to be protected. TPM parameter encryption using TPM sessions encrypts only the first (sized) parameter of a command or response.

End of informative comment

4.3 Remotely-Attested Channel

Start of informative comment

 © TCG 2025

This specification defines a channel that can be characterized as "remotely-attested" rather than "mutually-authenticated". Both the TPM and host use signing keys associated with their identities to establish their SPDM channel. However, at the time the channel is established, the TPM and host take their peer's public keys at face value. The TPM will not attempt to verify that a host is the correct host: rather, when evaluating certain policy assertions, the TPM will compare the host’s SPDM signing key in the policy with the key used to establish the current SPDM session. Likewise, the host will simply record the TPM's SPDM public key.

Later, an external verifier will examine evidence of which keys were used to establish the channel, and at that time will associate identities with those keys to determine whether the keys are trustworthy.

Therefore, as identities are not being verified locally as part of channel establishment, this specification characterizes the channel as "remotely-attested" rather than "mutually-authenticated."

End of informative comment

 © TCG 2025

5 TPM SPDM SECURE TRANSPORT PROTOCOL

This specification defines a secure transport protocol to protect the communications between a TPM and a host. The protocol, which is based on SPDM messages, is referred to within this specification as “TPM SPDM secure transport protocol”.

5.1 SPDM Specifications

This TPM SPDM secure transport protocol is based on the following specifications defined by DMTF:

- DSP0274 – Security Protocol and Data Model (SPDM) Specification version 1.3 or later [1]
- DSP0277 – Secured Messages using SPDM Specification version 1.2 [6]

Start of informative comment

SPDM is a protocol for performing message exchanges between devices. SPDM provides mutually-authenticated session key exchange protocols that enable confidentiality with integrity protected data communication.

DSP0277 defines a methodology for communicating application data securely by utilizing SPDM.

End of informative comment

5.2 TPM Supported SPDM Functionality

This TPM SPDM secure transport protocol supports only a subset of the SPDM messages defined in the SPDM specification [1] to establish a secure channel: plus, it adds TCG “vendor-defined” messages to:

- Exchange public keys (as the SPDM specification defines only messages to exchange digests and certificate chains)
- Transmit protected TPM commands within an SPDM secure session (apart from H-CRTM signals, see section 5.7.2.1.1)

This TPM SPDM secure transport protocol does not support the exchange of certificates and SPDM measurements defined in [1]. All measurement related fields in SPDM messages are set to zero or are absent.

Start of informative comment

The SPDM specification allows the use of raw public keys instead of certificate chains. Using raw public keys has the advantage that a TPM does not need to verify an X.509 certificate chain and it also allows the TPM signing key (used in the SPDM handshake) to be changed. Otherwise, if the key changes, the TPM would need the ability to issue a new X.509 certificate for it.

The SPDM specification defines a “vendor-defined” request and response message, which allows vendors or standards bodies to define their own SPDM messages. This TCG specification specifies TCG-defined messages in terms of “vendor-defined” messages.

End of informative comment

5.3 SPDM Secure Session Layer

This TPM SPDM secure transport protocol can be implemented in a dedicated SPDM secure session layer, which is mostly self-contained with only small changes to the other protocol layers.

“SPDM secure session layer” is used within this specification to describe the layer that is responsible for the SPDM session handling. The actual implementation of the SPDM secure session layer is outside the scope of this specification. The SPDM secure session layer SHALL be physical-interface-agnostic, i.e., support for SPDM messages is independent of the TPM physical interface (e.g., SPI or I2C).

The SPDM secure session layer implements the SPDM messages defined in section 5.5 and is responsible for:

 © TCG 2025

- Establishing a secure session
- Decrypting incoming data and forwarding it to the TPM application layer for processing
- Encrypting outgoing data and forwarding it to the transport layer for transmission

Figure 1 illustrates the high-level protocol layers in a TPM implementation with an SPDM secure session layer.

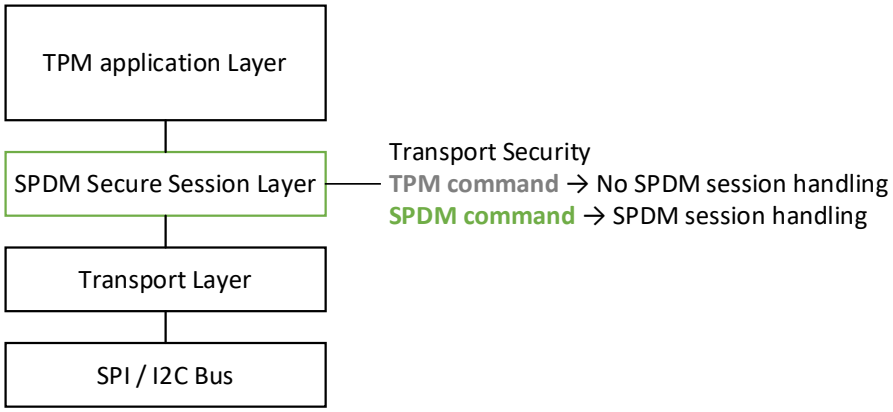


Figure 1: Protocol layers in a TPM implementation with an SPDM layer

Table 1 provides an overview of the TCG specifications that specify the functionality of the different TPM protocol layers illustrated in Figure 1.

TPM protocol layer	TCG specification	Description
TPM application layer	TPM Library specification [2] Platform TPM Profile (e.g. PTP [7])	Provides TPM capabilities (implements TPM commands)
SPDM secure session layer	TPM Communication over SPDM Secure Session (this document)	Provides secure channel capabilities (implements SPDM messages)
Transport layer	Platform TPM Profile (e.g. PTP [7])	Transport protocol
SPI / I2C bus	Platform TPM Profile (e.g. PTP [7])	Physical interface

Table 1: Applicability of TCG specifications

5.3.1 Architecture of a TPM Implementation with an SPDM Layer

Figure 2 illustrates the architecture of a TPM implementation with an SPDM secure session layer. The SPDM secure session layer interacts with the TPM application layer via internal Protected Capabilities defined in the TPM Library specification [2].

The SPDM secure session layer passes information to the TPM application layer via TPM Internal Protected Capabilities to enable the TPM application to take action on that information, such as evaluating that information in a policy assertion or setting the contents of TPM_CAP properties (TPM shielded locations).

 © TCG 2025

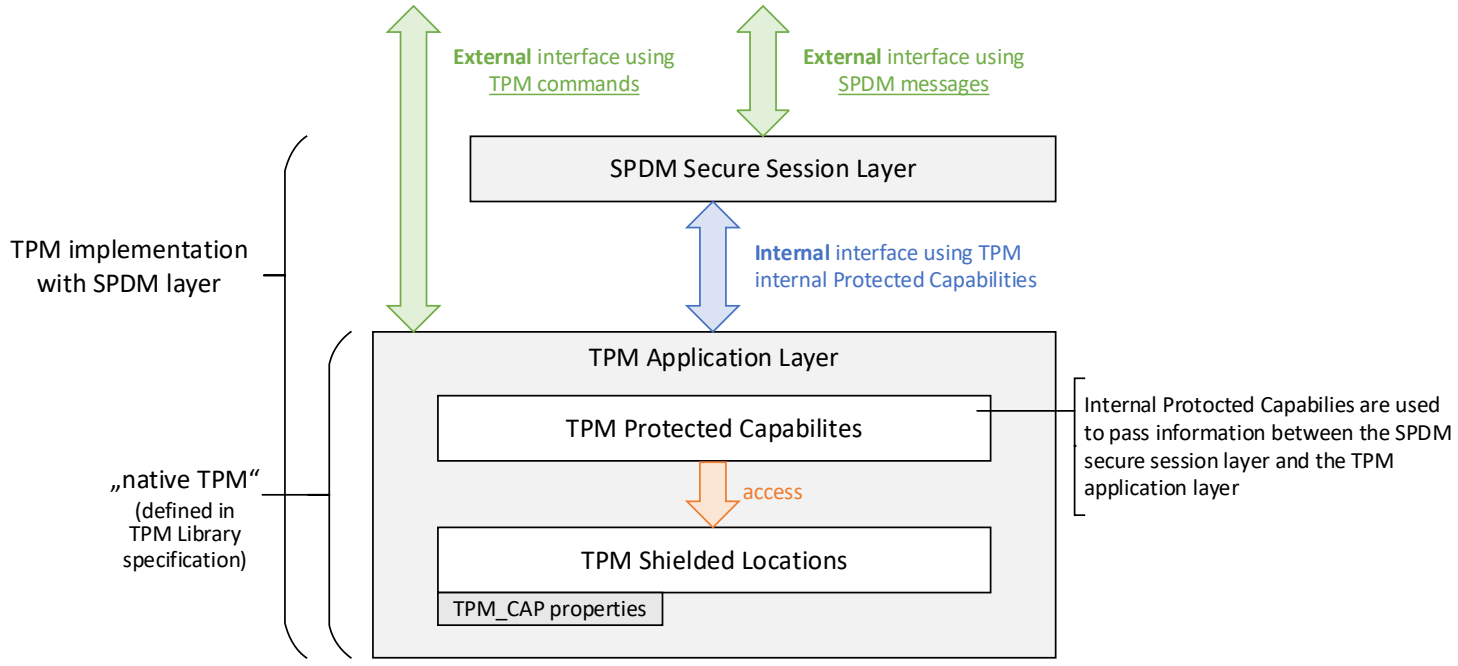


Figure 2: TPM implementation with SPDM layer architecture

5.4 Backward Compatibility

5.4.1 TPM Command Acceptance

This specification enables a TPM implementation with an SPDM secure session layer to be backward compatible with host implementations that do not support SPDM sessions.

A TPM implementation with an SPDM secure session layer SHALL accept regular TPM commands, which are not encapsulated in an SPDM message, at any time, independent of whether an SPDM session is active or not.

Start of informative comment

For a remotely-attested channel between a TPM and a host, there is limited security value in requiring the TPM to reject unencapsulated TPM commands once an SPDM session is active (i.e., requiring the TPM to accept only TPM commands protected by an SPDM session), since any Requester – including an attacker – can set up an SPDM session with the TPM. The Requester public key is provided to the TPM during the session handshake: therefore, the TPM does not know whether the public key belongs to a trustworthy SPDM Requester (host) or not (this is verified by an external verifier later).

However, prohibiting unencapsulated TPM commands once an SPDM session is active would impact backward compatibility of TPM host software (such as BIOS or OS), since it would require all software components running on a platform to be SPDM-capable (i.e., if one component has set up an SPDM session, all components would need to do so).

End of informative comment

5.4.2 FIFO/ CRB Interface and Locality

Start of informative comment

The byte stream of an SPDM message is sent over the same FIFO or CRB interface (defined in [7]) as the byte stream of a regular TPM command. SPDM messages are differentiated from TPM commands via a tag (see section 5.6.2). For compatibility, the (command/response) size is encoded at the same position, byte 3-6, as for a TPM command (see section 5.6).

 © TCG 2025

The locality of an SPDM request and the TPM command it encapsulates is determined by the address range in the FIFO or CRB register space, from which the SPDM request is sent (the same as for a regular TPM command).

This TPM SPDM secure transport protocol does not require any changes to the FIFO/ CRB interface and TPM HW registers.

End of informative comment

5.5 SPDM Messages

5.5.1 SPDM Requests and Responses

This section defines the subset of the (standard) SPDM messages defined in the SPDM specification [1] that are required for the TPM SPDM secure transport protocol.

A TPM's SPDM secure session layer SHALL implement the SPDM request and response messages listed in Table 2. SPDM messages that are specified in [1], but not included in Table 2 are not required by this specification and are Optional. Additional vendor-defined SPDM messages are described in section 5.5.2.

SPDM request message	SPDM response message
Capability discovery and negotiation	
GET_VERSION	VERSION
GET_CAPABILITIES	CAPABILITIES
NEGOTIATE_ALGORITHMS	ALGORITHMS
Session key establishment	
KEY_EXCHANGE	KEY_EXCHANGE_RSP
FINISH	FINISH_RSP
Session termination	
END_SESSION	END_SESSION_ACK
Error handling	
-	ERROR
Vendor-defined	
VENDOR_DEFINED_REQUEST	VENDOR_DEFINED_RESPONSE

Table 2: Mandatory SPDM commands

Start of informative comment

ERROR:

Note that if an SPDM command fails, the SPDM Responder sends an ERROR response message instead of the response message corresponding to the request. In this case, the SPDM Requester may resend the request, depending on the error code.

KEY_UPDATE:

This TPM SPDM secure transport protocol does not require support for the KEY_UPDATE command, which updates the session keys and resets the sequence number. The reason is that the length of the sequence number is sufficiently long (64-bit, see Table 20) that the sequence number is expected to never overflow (repeat).

Determine supported capabilities:

 © TCG 2025

Which SPDM commands are supported by a Requester and Responder are reported in the *Flags* field of the GET_CAPABILITIES request and CAPABILITIES response.

Determine SPDM support:

Whether a TPM supports SPDM functionality can be determined by:

- The availability of the TPM_CAP_SPDM_SESSION_INFO capability (readable via TPM2_GetCapability) (a TPM returns TPM_RC_VALUE if a capability is not implemented), or
- Sending an SPDM (GET_VERSION) request message (a TPM returns TPM_RC_BAD_TAG if SPDM is not supported because the SPDM message tags will not be recognized)

Note that GET_VERSION, depending on the connection handle, might reset a connection (see section 5.7.1).

End of informative comment

The following sub-sections define the requirements of this TPM SPDM secure transport protocol for each SPDM message.

5.5.1.1 GET_VERSION

A TPM's SPDM secure session layer SHALL support SPDM version 1.3 or later.

Start of informative comment

There are no relevant changes from SPDM version 1.2 to 1.3 for the TPM SPDM secure transport protocol defined in this specification. SPDM version 1.3 (which is the latest version at time of writing) adds optional support for "Extended capabilities" (see the description in GET_CAPABILITIES). This specification mandates no functionality from SPDM version 1.3. However, version 1.3 is defined as the minimum version to encourage consistent implementations across different vendors and limit the number of versions a host has to support.

Note that GET_VERSION may be sent by the Requester at any time to reset a connection (e.g., if the Requester has lost the session keys).

According to the SPDM specification [1] ("Table 6 – SPDM request and response messages validity"), GET_VERSION is only allowed to be sent outside of a session. Accordingly, GET_VERSION has to be sent as an SPDM clear message: it is not permitted to be sent as an SPDM secured message within a session. (For the SPDM clear and secured message format, see section 5.6.)

End of informative comment

5.5.1.2 GET_CAPABILITIES

This TPM SPDM secure transport protocol requires only a subset of the capabilities defined in the SPDM specification [1].

A TPM's SPDM secure session layer SHALL support the following SPDM capabilities:

- Message encryption and authentication in a secure session, including in the session handshake phase (i.e., ENCRYPT_CAP, MAC_CAP)
- The KEY_EXCHANGE message (i.e., KEY_EX_CAP)
- Mutual authentication (i.e., MUT_AUTH_CAP) (see section 5.5.1.2.1)
- (Vendor-defined) Raw public keys (i.e., PUB_KEY_ID_CAP)

Accordingly, a TPM's SPDM secure session layer SHALL set the flags in the *Flags* field of the CAPABILITIES response as indicated in Table 3.

 © TCG 2025

Capability Flag	Value
ENCRYPT_CAP	1
MAC_CAP	1
HANDSHAKE_IN_THE_CLEAR_CAP	0
KEY_EX_CAP	1
MUT_AUTH_CAP	1
CERT_CAP	0
PUB_KEY_ID_CAP	1

Table 3: Setting of SPDM Capabilities

Other capabilities specified in [1], but not included in Table 3 are not required by this specification and are Optional.

A TPM's SPDM secure session layer SHALL return an ERROR message of ErrorCode=InvalidPolicy (in response to the GET_CAPABILITIES request) if any of the ENCRYPT_CAP, MAC_CAP, KEY_EX_CAP or PUB_KEY_ID_CAP flags in the *Flags* field of the GET_CAPABILITIES request is not SET (1).

Note: A TPM's SPDM secure session layer is not required to check the settings of capability flags that it does not support.

Start of informative comment

The SPDM specification [1] requires that either CERT_CAP or PUB_KEY_ID_CAP must be SET (i.e., either certificate chains or raw public keys must be supported) if KEY_EX_CAP is SET. In this TCG specification, PUB_KEY_ID_CAP indicates support for the GET_PUB_KEY and GIVE_PUB_KEY messages, which are used to exchange the raw public keys during the session handshake (rather than that the public keys are provisioned).

SPDM version 1.3 defines "Extended capabilities", which allow the Responder to report its supported algorithms in the CAPABILITIES response. However, this requires support for the "Large SPDM message transfer mechanism" (indicated via CHUNK_CAP). This capability is optional in SPDM and is also not required for the TPM SPDM secure transport protocol.

End of informative comment

5.5.1.2.1 MUT_AUTH_CAP

Since the channel described in this specification is remotely-attested rather than mutually-authenticated (see section 4.3), the terms "mutual and Responder-only *signature verification*" are used instead of "mutual and Responder-only *authentication*". (Instead of exchanging certificates or provisioning trusted public keys as defined in the SPDM specification, the signature public keys are exchanged during the session handshake.)

A TPM's SPDM secure session layer SHALL support mutual and Responder-only signature verification.

The Requester selects the signature verification type via the MUT_AUTH_CAP flag in the GET_CAPABILITIES request.

If MUT_AUTH_CAP is SET in the GET_CAPABILITIES request, then a TPM's SPDM secure session layer SHALL:

- Set Bit 0 in the *MutAuthRequested* field in the KEY_EXCHANGE_RSP response to SET
- Return an ERROR message of ErrorCode=UnexpectedRequest in response to FINISH if the GIVE_PUB_KEY request (see section 5.5.2.4) was not received

If MUT_AUTH_CAP is CLEAR in the GET_CAPABILITIES request, then a TPM's SPDM secure session layer SHALL:

- Set Bit 0 in the *MutAuthRequested* field in the KEY_EXCHANGE_RSP response to CLEAR
- Return an ERROR message of ErrorCode=UnexpectedRequest if the GIVE_PUB_KEY request (see section 5.5.2.4) is received

 © TCG 2025

Table 4 summarizes the differences between mutual and Responder-only signature verification with respect to the different messages.

Signature verification type	**Value of** MUT_AUTH_CAP **in** GET_CAPABILIITES **request**	**Value of *MutAuthRequested*, Bit 0 in** KEY_EXCHANGE_RSP **response**	GIVE_PUB_KEY **request**	FINISH **request**
Mutual signature verification	SET	SET	Present (includes the Requester's public key)	Includes the Requester's signature
Responder-only signature verification	CLEAR	CLEAR	Absent	Does not include the Requester's signature

Table 4: Mutual and Responder-only signature verification

Start of informative comment

Responder-only authentication:

In the SPDM specification [1], Responder-only authentication is mandatory, while mutual authentication is optional. Whether a Requester or Responder supports mutual authentication is reported in the MUT_AUTH_CAP flag. The SPDM specification does not define a flag to indicate support for Responder-only authentication.

MutAuthRequested:

The Requester's reporting of MUT_AUTH_CAP has no impact on the TPM's SPDM secure session layer's reporting of MUT_AUTH_CAP, but it determines the TPM's SPDM secure session layer's setting of the *MutAuthRequested* field in the KEY_EXCHANGE_RSP response. As defined in [1], *MutAuthRequested* indicates whether the Responder requires the Requester to authenticate.

Encapsulated messages:

This TPM SPDM secure transport protocol does not support an encapsulated request flow or an optimized encapsulated request flow for mutual authentication as defined in [1]. The Requester public key is provided to the Responder in a single (TCG vendor-defined GIVE_PUB_KEY) request message for simplicity and is not exchanged using GET_ENCAPSULATED_REQUEST and DELIVER_ENCAPSULATED_RESPONSE messages.

End of informative comment

5.5.1.2.2 CTExponent

The *CT* timing parameter specified in [1] SHALL apply only to the standard SPDM request messages (see section 5.5.1) but not to the TCG vendor-defined SPDM request messages (see section 5.5.2).

A TPM's SPDM secure session layer SHALL set the *CTExponent* field in the CAPABILITIES response to a value that is sufficiently large to accommodate the processing time of any standard SPDM request message, so that the TPM can always respond (to a standard SPDM request message) within the *CT* time period and never has to return an ERROR message of ErrorCode=ResponseNotReady.

Start of informative comment

This TPM SPDM secure transport protocol does not require support for the RESPOND_IF_READY command, which is used to retrieve the response (to the original request) upon receipt of the ResponseNotReady error code. The reason is that it is not desirable for a TPM to abort ongoing operations in order to return this error.

As specified in [1], for request messages that perform cryptographic operations, if a Responder cannot respond within the required timing (specified as timing parameter *CT*), it can respond with a ResponseNotReady error code to indicate that it needs more time, where $CT = 2^{CTExponent}$ microseconds (µs).

 © TCG 2025

Note that a Responder that does not support the RESPOND_IF_READY command must not return the ResponseNotReady error code.

End of informative comment

5.5.1.2.3 DataTransferSize

Start of informative comment

It is recommended that a Requester sets the *DataTransferSize* field in the GET_CAPABILITIES request to at least the maximum TPM response size (MAX_RESPONSE_SIZE, which can be read via the TPM property TPM_PT_MAX_RESPONSE_SIZE) plus the overhead for the VENDOR_DEFINED_RESPONSE message (which is 17 bytes, see Figure 12). This avoids a situation in which the TPM's SPDM secure session layer responds with an ERROR message of ErrorCode=ResponseTooLarge if the size of a response message exceeds the size of the *MaxSPDMmsgSize* of the Requester.

As specified in [1], *DataTransferSize* indicates the maximum buffer size of the Requester or Responder for receiving a single and complete SPDM message. Note that *DataTransferSize* excludes transport headers, encryption headers, and MAC (meaning it indicates the size of the SPDM Clear message). This (TCG) specification does not require support for the Large SPDM message transfer mechanism (i.e., transferring a message in chunks), in which case the *MaxSPDMmsgSize* field is equal to the *DataTransferSize*.

If an ERROR message of ErrorCode=ResponseTooLarge is received (in response to TPM_CMD) (e.g., if the above recommendation is not followed by a Requester), it can be assumed that the TPM's application layer has successfully processed the TPM command (i.e., only the TPM's SPDM secure session layer failed to return the TPM response) when handling this error case.

End of informative comment

5.5.1.3 NEGOTIATE_ALGORITHMS

5.5.1.3.1 SPDM Algorithms

A TPM's SPDM secure session layer SHALL support the algorithm set B or C listed in Table 5 (that is, support an algorithm set with a security strength of minimum 192 bits).

A TPM's SPDM secure session layer MAY support additional algorithms specified by the SPDM specification [1] (see section 5.5.1.3.1.1).

For compatibility, a TPM's SPDM secure session layer SHOULD NOT support extended algorithms.

Algorithm Set	**Security strength**	**Signature generation and verification**	**Hash** (including HMAC)	**ECDHE**	**AEAD**
A	128 bits	ECDSA NIST P-256	SHA-256	secp256r1 (NIST P-256)	AES-128-GCM
B	192 bits	ECDSA NIST P-384	SHA-384	secp384r1 (NIST P-384)	AES-256-GCM
C	256 bits	ECDSA NIST P-521	SHA-512	secp521r1 (NIST P-521)	AES-256-GCM

Table 5: SPDM algorithm sets

If a host and its TPM's SPDM secure session layer support multiple key sizes for the same algorithm, the method used by the TPM's SPDM secure session layer to select the key size is vendor-specific. It is recommended that a TPM's SPDM secure session layer selects only algorithms with an equal security strength (with the exception of AES-256, which is used instead of AES-192).

Start of informative comment

The Responder selects the algorithms based on the Requester-supported algorithms (indicated in NEGOTIATE_ALGORITHMS). According to [1], the Responder does not return an error if no algorithm for a

 © TCG 2025

primitive is selected: instead, no bits in the corresponding bitmap are set and the Responder should request resynchronization.

ECDSA, SHA-2 and AES are selected as algorithms since they are supported by the TPM Library specification [2] and required by the PC Client Platform TPM Profile (PTP) [7] (at time of writing).

End of informative comment

5.5.1.3.1.1 SPDM Specification Supported Algorithms

Start of informative comment

The SPDM specification [1] provides a wide choice of algorithms for each cryptographic operation:

- Asymmetric signature generation and verification:
  - RSA 2k, 3k, 4k (RSASSA or RSAPSS)
  - **ECDSA NIST P-256, NIST P-384, NIST P-521**
  - SM2 P256
  - EdDSA ed25519, ed448
- Hash:
  - **SHA-256, SHA-384, SHA-512**
  - SHA3-256, SHA3-384, SHA3-512
  - SM3-256
- DHE:
  - ffdhe2048, ffdhe3072, ffdhe4096
  - **secp256r1, secp384r1, secp521r1 (NIST P-256, NIST P-384, NIST P-521)**
  - SM2 P256
- AEAD:
  - **AES-128-GCM, AES-256-GCM**
  - CHACHA20-POLY1305
  - AEAD-SM4-GCM
- Key schedule (key derivation function):
  - **SPDM key schedule** (defined in [1], section 12 Key schedule)

Note that for the key schedule algorithm, only one choice is defined.

The algorithms selected for the algorithm sets in Table 5 are marked in **bold** in the above list.

In addition to the above algorithms, the SPDM specification allows extended algorithms, which may be specified by a vendor or standards body.

End of informative comment

5.5.1.3.1.2 Algorithm Selection

Start of informative comment

ECC NIST-P384, SHA-384 and AES-256 are selected as the minimum Mandatory algorithms for this TPM SPDM secure transport protocol because these are the strongest algorithms that are already supported by TPMs, which are compliant with the PC Client TPM Platform Profile 1.06 [7].

Based on the selected set of algorithms, this TPM SPDM secure transport protocol uses the following cryptographic primitives/ functions:

- ECDSA signature generation and verification
- ECC key agreement (ECC key generation + ECDH)
- SHA
- HMAC
- HKDF (defined in RFC 5869 [8], compliant with NIST SP 800-56Cr2)
- AES-GCM

 © TCG 2025

ECC is selected as the mandatory asymmetric algorithm because ECC is required for the computation of the DHE secret. The TPM does not support the Finite Field Diffie-Hellman Ephemeral Parameters: ffdhe2048, ffdhe3072, ffdhe4096.

End of informative comment

5.5.1.3.2 Opaque Data Format

A TPM's SPDM secure session layer SHALL use the "general opaque data format" defined by the SPDM specification [1] for the format of the *OpaqueData* field (which is included in KEY_EXCHANGE, see section 5.5.1.4.2).

A TPM's SPDM secure session layer SHALL:

- Return an ERROR message of ErrorCode=InvalidRequest if the *OpaqueDataFmt1* bit in the *OtherParamsSupport* field of the NEGOTIATE_ALGORITHMS request is not SET (1)
- Set the *OpaqueDataFmt1* bit in the *OtherParamsSelection* field of the ALGORITHMS response to SET (1).

5.5.1.4 KEY_EXCHANGE

5.5.1.4.1 Slot ID

Start of informative comment

This TPM SPDM secure transport protocol does not exchange certificates: therefore, the *slotID* fields are not used to reference the slot of a certificate chain. Instead, the *slotID* fields are set to special values (0xFF and 0xF) defined by the SPDM specification [1] to indicate that the public keys are exchanged through other means (namely, via the vendor-defined messages GET_PUB_KEY and GIVE_PUB_KEY).

End of informative comment

A TPM's SPDM secure session layer SHALL:

- Return an ERROR message of ErrorCode=InvalidRequest if the slot ID in the *Param2* field in the KEY_EXCHANGE request is not set to 0xFF, and
- Set the slot ID in the *SlotIDParam* field in the KEY_EXCHANGE_RSP response to 0xF.

5.5.1.4.2 Opaque Data

Following DSP0277 [6], "Secured Messages using SPDM specification", the *OpaqueData* fields in the KEY_EXCHANGE messages are used to negotiate the format for the SPDM secured messages. For this, both endpoints negotiate a Secured Message (transport binding) version (see section 5.5.1.4.2.1).

The *OpaqueData* field in the KEY_EXCHANGE request and KEY_EXCHANGE_RSP response SHALL be encoded in the "general opaque data format" as defined in the SPDM specification [1]. The *OpaqueData* field SHALL contain one "Secured Message opaque element" as defined in DSP0277 [6].

Start of informative comment

DSP0277 does not comprehensively define all requirements related to secured messages: instead, it lists some requirements, which have to be defined by a "transport binding specification".

This TCG (TPM Communication over SPDM Secure Session) specification serves as a "transport binding specification". DSP0277 defines variable-length secured message fields for which the lengths are defined in section 5.6.4 of this (TCG) specification.

OpaqueData is used to negotiate a Secured Message version, which is defined in the transport binding specification, that both Requester and Responder support.

As depicted in Table 6, the content of the "SMData" field in the Secured Message opaque element is different for a Requester and Responder:

 © TCG 2025

- The Requester provides a list of Requester-supported versions in *SecuredMsgVers* in the request, (following the format defined in [6], section 8.1.2, “Supported version list data format”).
- The Responder selects one common version that is supported by both, which is returned in *SelectedVersion* in the response (following the format defined in [6], section 8.1.1, “Version selection data format”).

Field	Requester	Responder
SMDataID	1 (Supported version list)	0 (Version selection)
SMData	SecuredMsgVers	SelectedVersion (see Table 7)

Table 6: Secured Message data

End of informative comment

5.5.1.4.2.1 Secured Message Version

A TPM’s SPDM secure session layer SHALL support the Secured Message (SM) version defined in Table 7, which identifies the Secured Message format defined in this version of the specification (see section 5.6.5).

Secured Message Version	Value
MajorVersion	1
MinorVersion	0
UpdateVersionNumber	Not used (set to 0)
Alpha	Not used (set to 0)

Table 7: Secured Message version

A TPM’s SPDM secure session layer SHALL return an ERROR message of ErrorCode=Vd_SMVersionMismatch if there is no common Secured Message version that is supported by both endpoints.

Start of informative comment

The format of the Secured Message (SM) version is defined in DSP0277 [6], section 6 “Version support”.

Because this (TCG) specification defines more than just the Secured Message format for the TPM SPDM secure transport protocol, the Secured Message version is defined as an independent version number, rather than the version of this specification. If a future version of this specification requires a change to the Secured message format, the SM version will be incremented to identify that change.

The negotiation will fail if no common SM version is supported (just as it fails if end points have no commonality in other negotiated settings, such as SPDM version and algorithms).

End of informative comment

5.5.1.4.3 Mutual Signature Verification

See section 5.5.1.2.1 MUT_AUTH_CAP on how to set “Bit 0” in the *MutAuthRequested* field in the KEY_EXCHANGE_RSP response.

5.5.1.5 FINISH

5.5.1.5.1 Slot ID

Just as for KEY_EXCHANGE, a TPM’s SPDM secure session layer SHALL return an ERROR message of ErrorCode=InvalidRequest if the slot ID in the *Param2* field of the FINISH request is not set to 0xFF.

5.5.1.5.2 Mutual Signature Verification

See section 5.5.1.2.1 MUT_AUTH_CAP for whether the *Signature* field in the FINISH request is required to be present or not.

 © TCG 2025

5.5.1.5.3 ResponderVerifyData

The session handshake phase in this TPM SPDM secure transport protocol is required to be encrypted (the TPM's SPDM secure session layer sets HANDSHAKE_IN_THE_CLEAR_CAP = 0 as specified in section 5.5.1.2), therefore, the *ResponderVerifyData* field (HMAC signature) in the FINISH_RSP response SHALL be absent as defined in the SPDM specification [1].

5.5.1.6 END_SESSION

No special considerations are required for these messages.

Start of informative comment

The END_SESSION_ACK response is sent within a session (as an SPDM secured message).

End of informative comment

5.5.1.7 ERROR

Error conditions and error codes for SPDM messages are specified in [1].

This specification defines additional error conditions, which are consolidated in Table 8. These use either SPDM standard or TCG vendor-defined error codes (highlighted in blue). The format and value of TCG vendor-defined error codes is specified in section 5.5.1.7.1.

For some errors specified in [1], the SPDM specification allows a Responder to either silently discard a request or respond with a defined error code. A TPM's SPDM secure session layer SHALL respond with an ERROR message (rather than silently discard a request).

A TPM with an SPDM secure session layer SHALL nullify the transcript hash on any error that occurs between GET_VERSION and FINISH_RSP (i.e., messages that are included in the transcript hash computation). In response to the failed request message, a TPM with an SPDM secure session layer SHALL return an ERROR message with an error code that indicates the actual error (not RequestResynch). In response to all subsequent requests within this connection, a TPM with an SPDM secure session layer SHALL return an ERROR message of ErrorCode=RequestResynch (unless the request is GET_VERSION).

A TPM with an SPDM secure session layer is not required to support a retry of request messages.

Start of informative comment

An error might occur either before or after the (request or response) message was extended to the transcript hash. For simplicity, any error that occurs until the session is established nullifies the transcript hash. (The transcript hash is used to derive the session keys and is only needed until the session is established.)

Returning the RequestResynch error code only in the next request allows error information to be conveyed to the Responder.

If a retry of request messages is not supported, a Responder will return an ERROR message of ErrorCode=UnexpectedRequest (for a retried message).

The SPDM specification [1] requires that:

- DecryptError terminates the session
- RequestResynch requires the connection to be reset with GET_VERSION

End of informative comment

Error code returned in SPDM ERROR response message	**Returned in response to**	**Description**	**Referenced in section**

 © TCG 2025

InvalidPolicy	GET_CAPABILITIES	The Requester does not support the required capabilities (ENCRYPT_CAP, MAC_CAP, KEY_EX_CAP, and PUB_KEY_ID_CAP) are required to be SET)	5.5.1.2
UnexpectedRequest	FINISH	Mutual signature verification is required but GIVE_PUB_KEY was skipped	5.5.1.2.1
UnexpectedRequest	GIVE_PUB_KEY	No Mutual signature verification is required but GIVE_PUB_KEY was sent	5.5.1.2.1
InvalidRequest	NEGOTIATE_ALGORITHMS	An incorrect Opaque data format was selected (required to use “General opaque data format”)	5.5.1.3.2
InvalidRequest	KEY_EXCHANGE FINISH	Incorrect Slot ID (required to be set to 0xFF)	5.5.1.4.1 5.5.1.5.1
Vd_SMVersionMismatch	KEY_EXCHANGE	No common format for SPDM secured message	5.5.1.4.2.1
Vd_InvalidCode	Any VENDOR_DEFINED_ REQUEST	The vendor-defined request contains an invalid *VdCode*	5.5.2.2
Vd_InvalidData	GIVE_PUB_KEY	The vendor-defined request contains invalid *VdData*: A Requester public key is not provided in a valid TPMT_PUBLIC structure	5.5.2.4 5.5.2.5
Vd_LocalityMismatch	TPM_CMD_L*	Locality of *VdCode* does not match the locality indicated by the address register	5.5.2.6
ResetRequired	Any SPDM message except for TPM_CMD and TPM_CMD_L*	The TPM is in Failure Mode This error terminates all connections.	5.5.3.2.1.1
Vd_UnsupportedTpmCmd	TPM_CMD, TPM_CMD_L*	The TPM command is not accepted within an SPDM session (allowed only for vendor-specific commands such as a Field Upgrade start command)	5.5.3.2.1.1
Vd_InvalidConnHandle	GET_VERSION	The TPM supports only one connection and *ConnectionHandle* is not zero	5.7.1
Vd_ConnLimitExceeded	GET_VERSION	The connection limit has been exceeded	5.7.1
Vd_InvalidConnHandle	Any SPDM message	No connection is associated with *ConnectionHandle*	5.7.1

Table 8: Error codes

(TCG Vendor-defined) Error code	Value	(TCG Vendor-defined) Error data
Vd_SMVersionMismatch	0x01	0x00
Vd_InvalidCode	0x02	0x00
Vd_InvalidData	0x03	0x00
Vd_LocalityMismatch	0x04	0x00
Vd_UnsupportedTpmCmd	0x08	0x00
Vd_InvalidConnHandle	0x10	0x00
Vd_ConnLimitExceeded	0x11	0x00

 © TCG 2025

Vd_TpmVendorDefined	0x2F	Defined by TPM vendor (1-byte)

Table 9: Vendor-defined error code values

Start of informative comment

The Vd_TpmVendorDefined error can be used for TPM vendor-specific error codes (with error data indicating the specific error condition).

This TCG specification will use error codes values in the range of 0x00-0x2F. It is recommended that other TCG specifications that also specify TCG vendor-defined error codes assign a distinct range to avoid potential collisions.

End of informative comment

5.5.1.7.1 Vendor-defined Error Format

The SPDM specification [1] defines a vendor or standards-defined *ErrorCode* (0xFF), which is supposed to be used by a vendor or standardization body to specify their own *ExtendedErrorData*.

The ERROR message for TCG-defined error codes SHALL use the following format as illustrated in Figure 3:

- *Param1* (*ErrorCode*) SHALL be set to "Vendor or Standards-Defined" (0xFF)
- *Param2* (Error data), which encodes the standards body ID, SHALL be set 0x1 (TCG)
- For *ExtendedErrorData,*
  - *Len* field SHALL be set to 0 (i.e., *VendorID* field is absent)
  - *OpaqueErrorData* SHALL be two bytes and contain a TCG-defined error code consisting of
    - VdErrorCode (1 byte)
    - VdErrorData (1 byte) (set to 0 if unused)

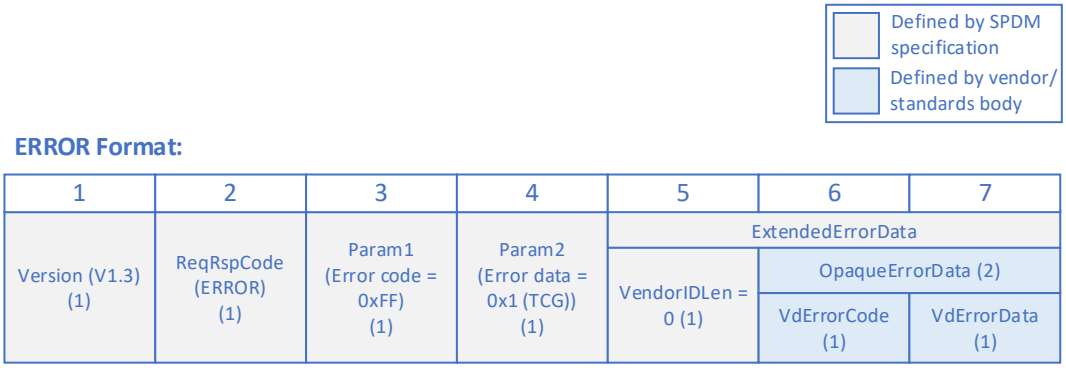


Figure 3: Vendor-defined ERROR

Start of informative comment

The 1-byte *VdErrorCode* and *VdErrorData* are defined analogous to the SPDM defined error code and data. Extended error data is not used for TCG vendor-defined error codes.

End of informative comment

5.5.2 Vendor-defined Requests and Responses

The SPDM specification [1] defines a vendor-defined request and response, which is supposed to be used by a vendor or standardization body to specify their own SPDM messages.

This section defines the vendor-defined SPDM messages specified by TCG, which are used in this TPM SPDM secure transport protocol.

 © TCG 2025

A TPM's SPDM secure session layer SHALL implement the vendor-defined SPDM request and response messages listed in Table 10 that are labeled as Mandatory.

SPDM request message	SPDM response message	Mandatory (M) / Optional (O)
Public key exchange		
GET_PUB_KEY	GET_PUB_KEY_RSP	M
GIVE_PUB_KEY	GIVE_PUB_KEY_RSP	M
TPM command transmission		
TPM_CMD	TPM_RSP	M
TPM_CMD_L0	TPM_RSP	O
TPM_CMD_L1	TPM_RSP	O
TPM_CMD_L2	TPM_RSP	O
TPM_CMD_L3	TPM_RSP	O
TPM_CMD_L4	TPM_RSP	O

Table 10: Mandatory vendor-defined SPDM commands

The TCG vendor-defined commands SHALL follow the request and response format of VENDOR_DEFINED_REQUEST request and VENDOR_DEFINED_REQUEST response (see Figure 12), where:

- *StandardID* field SHALL be set to 0x1 (TCG)
- *Len* field SHALL be set to 0 (i.e., *VendorID* field is absent)
- *VendorDefinedReqPayload* and *VendorDefinedRespPayload* fields depend on the message (see section 5.5.2.2)

Start of informative comment

Two commands are defined for the public key exchange (rather than exchanging the public keys in one single command) because this provides more flexibility. It enables uses cases where, for example the Requester signing key depends on the value of the TPM public key. It also allows a host to query the TPM public key for monitoring reasons without starting a session.

End of informative comment

5.5.2.1 Transcript Hash Considerations for *_PUB_KEY

Start of informative comment

The transcript hash is computed as specified in the SPDM specification [1], section 10.17.1 "Transcript and transcript hash calculation rules for KEY_EXCHANGE". The transcript includes the hash of the public key if a certificate is not used.

Following [1], the transcript hash does not include the complete GET_PUB_KEY and GIVE_PUB_KEY request and response messages. It includes only:

- The hash of the Responder public key (transmitted in GET_PUB_KEY) and
- The hash of the Requester public key (transmitted in GIVE_PUB_KEY) – for mutual signature verification only

End of informative comment

The hash of the public key (which is included in the transcript hash) SHALL be computed over the public key in its TPMT_PUBLIC big-endian format.

 © TCG 2025

5.5.2.2 Vendor-defined Request and Response Payload

This section defines the *VendorDefinedReqPayload* and *VendorDefinedRespPayload* fields for the vendor-defined messages (see Figure 12).

The *VendorDefinedReqPayload* and *VendorDefinedRespPayload* fields SHALL consist of:

- An 8-byte *VdCode* identifying the vendor-defined request and response
- An (optional) *VdData*, which is the actual request or response data

Table 11 specifies *VdCode* and *VdData* for the vendor-defined SPDM messages used by the TPM SPDM secure transport protocol.

Vendor-defined request or response	***VdCode* (big endian)** `hex ("ascii string")`	***VdData***	**Type (of *VdData*)**
GET_PUB_KEY	`0x4745545F5055424B ("GET_PUBK")`	-	-
GET_PUB_KEY_RSP	`0x4745545F5055424B ("GET_PUBK")`	Public key	TPMT_PUBLIC
GIVE_PUB_KEY	`0x474956455F505542 ("GIVE_PUB")`	Public key	TPMT_PUBLIC
GIVE_PUB_KEY_RSP	`0x474956455F505542 ("GIVE_PUB")`	-	-
TPM_CMD	`0x54504D325F434D44 ("TPM2_CMD")`	TPM command	uint8[] Note: max. size is MAX_COMMAND_SIZE
TPM_RSP	`0x54504D325F434D44 ("TPM2_CMD")`	TPM response	uint8[] Note: max. size is MAX_RESPONSE_SIZE
TPM_CMD_L0	`0x54504D32434D4430 ("TPM2CMD0")`	TPM command	uint8[]
TPM_CMD_L1	`0x54504D32434D4431 ("TPM2CMD1")`	TPM command	uint8[]
TPM_CMD_L2	`0x54504D32434D4432 ("TPM2CMD2")`	TPM command	uint8[]
TPM_CMD_L3	`0x54504D32434D4433 ("TPM2CMD3")`	TPM command	uint8[]
TPM_CMD_L4	`0x54504D32434D4434 ("TPM2CMD4")`	TPM command	uint8[]

Table 11: Vendor-defined request and response code and payload

A TPM's SPDM secure session layer SHALL return an ERROR message of ErrorCode=Vd_InvalidCode if an invalid *VdCode* is provided.

5.5.2.3 GET_PUB_KEY

Start of informative comment

 © TCG 2025

The GET_PUB_KEY command is used by the SPDM Requester to get the public key of the SPDM Responder. The Requester uses the Responder public key to verify the asymmetric signature transmitted in the KEY_EXCHANGE_RSP response message.

End of informative comment

A TPM's SPDM secure session layer SHALL return the TPM SPDM-identity public key using the object public structure (TPMT_PUBLIC) as defined in section 6.1 in the GET_PUB_KEY_RSP response.

5.5.2.4 GIVE_PUB_KEY

Start of informative comment

The GIVE_PUB_KEY command is used by the SPDM Requester to give its public key to the SPDM Responder. The Responder uses the Requester public key to verify the asymmetric signature transmitted in the FINISH request message.

End of informative comment

The GIVE_PUB_KEY command is required if mutual signature verification is required, otherwise, the command is required to be absent. The corresponding error return codes are defined in section 5.5.1.2.1 MUT_AUTH_CAP.

A TPM's SPDM secure session layer SHALL return an ERROR message of ErrorCode=Vd_InvalidData if the public key is not provided in a valid TPMT_PUBLIC structure or the algorithm identifiers do not match the negotiated algorithms.

A TPM's SPDM secure session layer SHOULD NOT verify any TPM object attributes.

5.5.2.5 TPM_CMD

The TPM_CMD command is used to transmit a TPM command within an SPDM message.

A TPM's SPDM secure session layer SHALL return the TPM response in the TPM_RSP response.

Start of informative comment

If *VdData* does not contain a valid TPM command, the TPM_RSP response contains a TPM error response (e.g., TPM_RC_BAD_TAG).

End of informative comment

5.5.2.6 TPM_CMD_L*

Start of informative comment

The TPM_CMD_L* requests are locality-aware versions of the TPM_CMD request. The intent is to provide integrity protection against an attacker that would modify the unprotected address register and thus manipulate the locality of the received TPM command.

The encapsulated TPM command and response is the same as in the TPM_CMD request and TPM_RSP response.

The vendor-defined request code (*VdCode*) of TPM_CMD_L* indicates the Requester's intended locality of an encapsulated TPM command (e.g., TPM_CMD_L0 for locality 0, TPM_CMD_L4 for locality 4). A TPM's SPDM secure session layer will expect a TPM command using the appropriate TPM_CMD_L* code to match the locality of the address register used when the command is sent.

End of informative comment

A TPM's SPDM secure session layer SHALL return an ERROR message of ErrorCode=Vd_LocalityMismatch if the locality indicated by a TPM_CMD_L* vendor-defined request code is different from the locality indicated by the address register of the received message.

 © TCG 2025

5.5.3 SPDM Message Flow

5.5.3.1 Message Flow Requirements

The message flow for the (standard) SPDM messages is defined in the SPDM specification [1]. This section defines additional requirements for vendor-defined SPDM messages – including the SPDM request and response that encapsulates (protects) a TPM command.

Figure 4 depicts the message flow with mutual signature verification, including the vendor-defined messages GET_PUB_KEY and GIVE_PUB_KEY.

Figure 5 depicts the message flow with Responder-only signature verification, where GIVE_PUB_KEY is omitted.

A more detailed flow is shown in Figure 6, which includes a TPM_CMD secured message sent after the handshake when the session has been established.

The following ordering rules apply to vendor-defined SPDM messages:

- GET_PUB_KEY **MUST** be sent immediately before KEY_EXCHANGE
- GIVE_PUB_KEY **MUST** be sent immediately after KEY_EXCHANGE – for mutual signature verification only
- TPM_CMD, TPM_CMD_L* **MUST** be sent after FINISH

Figure 4 and Figure 5 depict how the order of the SPDM messages is enforced by the Responder via states.

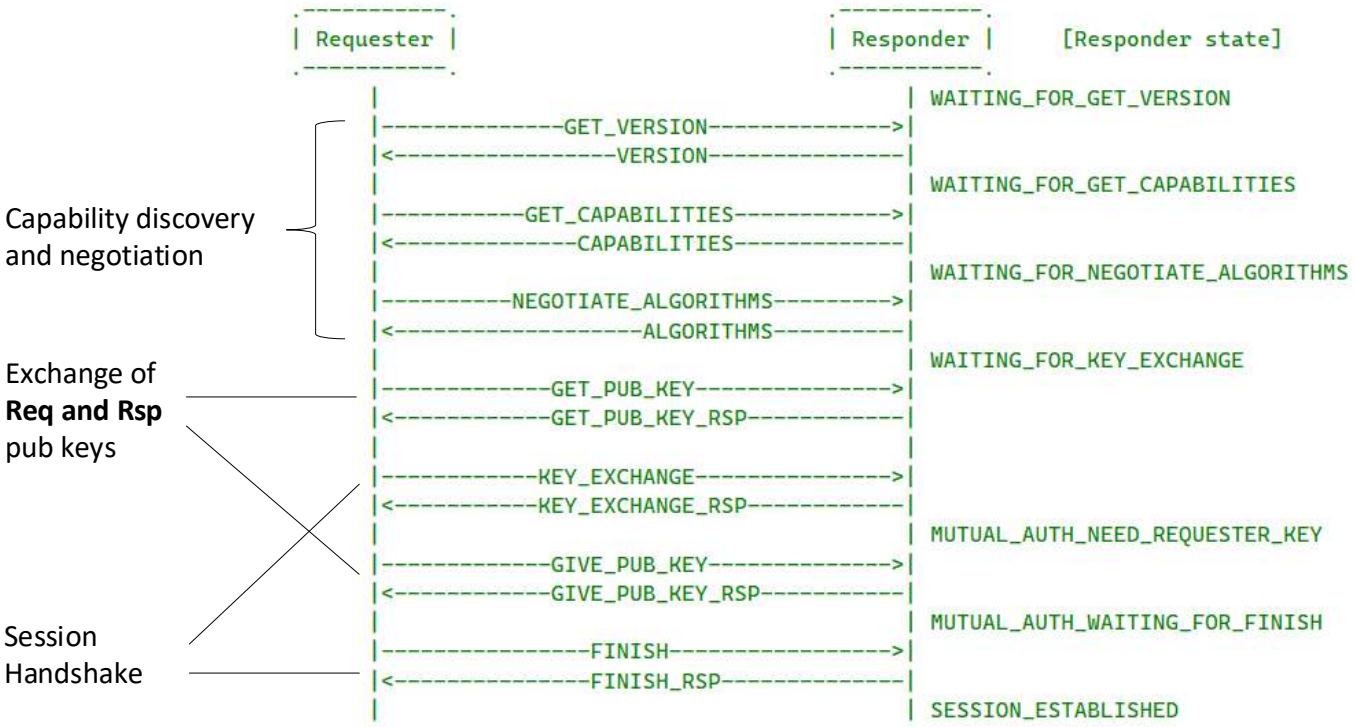


Figure 4: SPDM message flow with mutual signature verification

 © TCG 2025

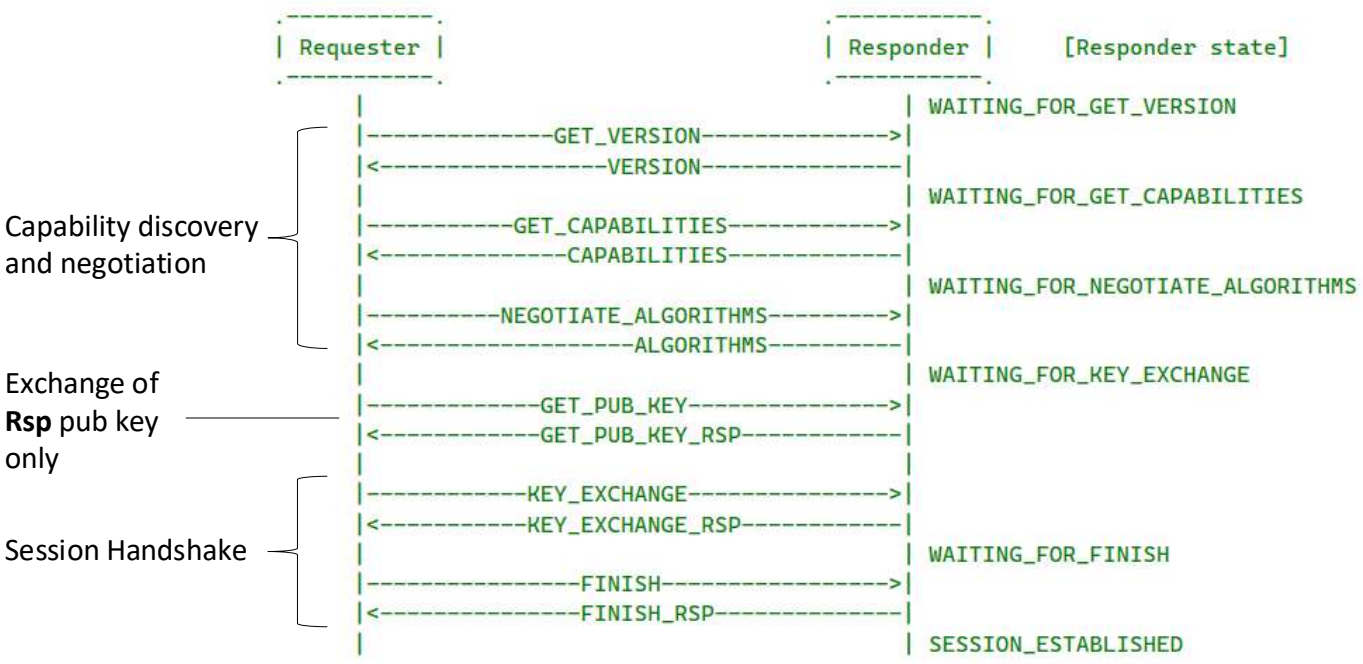


Figure 5: SPDM message flow with Responder-only signature verification

Figure 6 includes also the cryptographic data (keys and signatures) that are exchanged in the SPDM messages and depicts the different phases.

 © TCG 2025

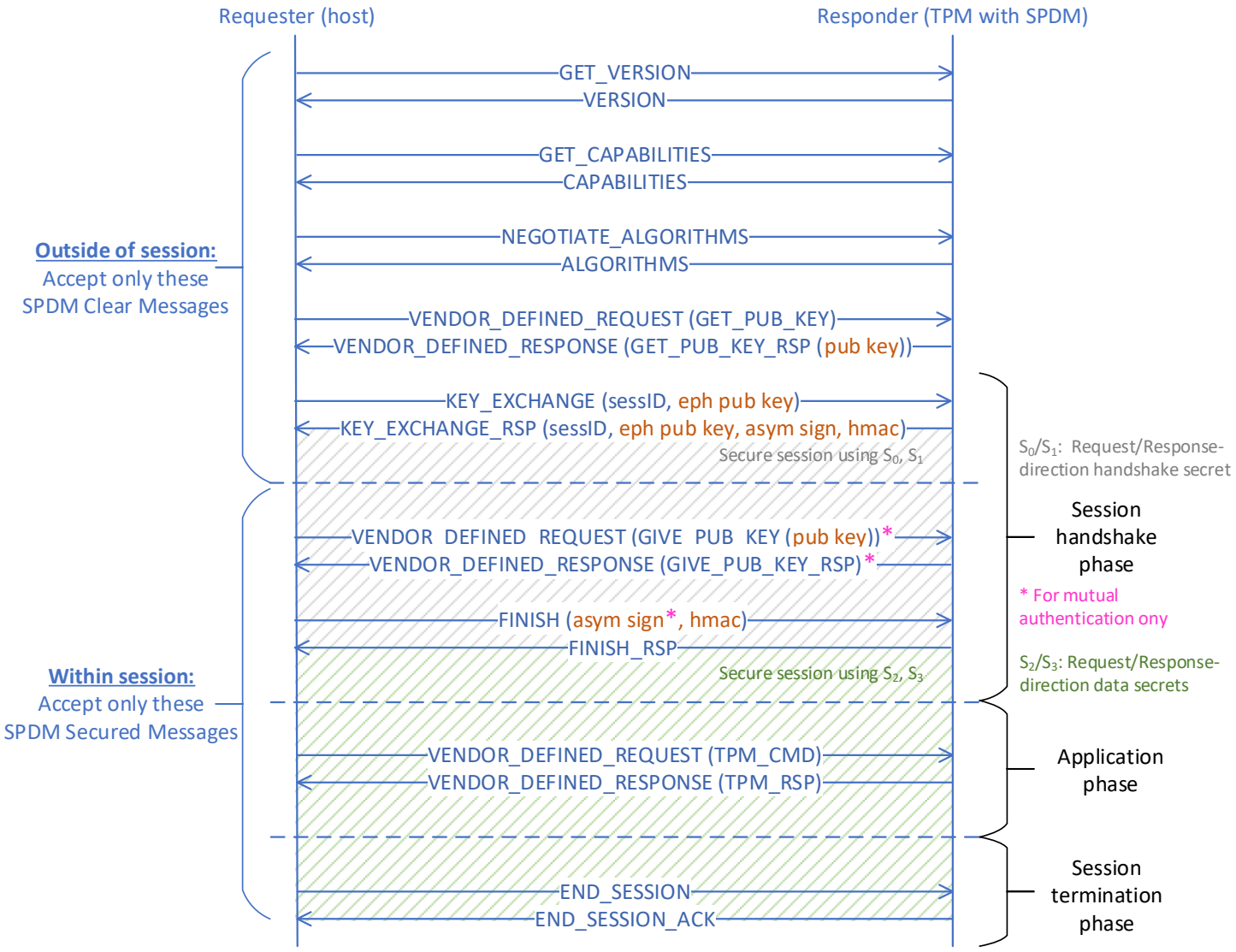


Figure 6: Detailed SPDM message flow

The following requirements apply to a TPM's SPDM secure session layer:

Outside of a session, a TPM's SPDM secure session layer SHALL accept only the following SPDM clear messages:

- GET_VERSION
- GET_CAPABILITIES
- NEGOTIATE_ALGORITHMS
- VENDOR_DEFINED_REQUEST (GET_PUB_KEY)
- KEY_EXCHANGE

Within a session, a TPM's SPDM secure session layer SHALL accept only the following SPDM secured messages:

- VENDOR_DEFINED_REQUEST (GIVE_PUB_KEY) – for mutual signature verification only
- FINISH
- VENDOR_DEFINED_REQUEST (TPM_CMD, TPM_CMD_L*) – zero or more messages
- END_SESSION

 © TCG 2025

For TPM commands accepted by a TPM implementation with an SPDM secure session layer, see section 5.4.1.

Start of informative comment

GET_VERSION may be sent at any time (outside of an SPDM session). An important feature of GET_VERSION is that it may be sent to reset a connection, e.g. if the host has lost the session keys. According to the SPDM specification [1] ("Table 6 – SPDM request and response messages validity"), GET_VERSION is only allowed to be sent outside of a session. Accordingly, GET_VERSION is always sent as SPDM clear message.

END_SESSION may be sent at any time within an SPDM session.

KEY_EXCHANGE may be sent within a connection to:

- Restart a session that has been terminated with END_SESSION, or
- Start a new session within a connection - if multiple sessions are supported

End of informative comment

5.5.3.1.1 SPDM Request and Response Messages Validity

Table 12 summarizes the conditions under which an SPDM request and response can be issued and is derived from "Table 6 — SPDM request and response messages validity" in the SPDM specification [1], but including just the commands used by this TPM SPDM secure transport protocol and extended with the vendor-defined messages defined by TCG. The vendor-defined messages (highlighted in blue) use the VENDOR_DEFINED_REQUEST and VENDOR_DEFINED_RESPONSE messages.

Request	**Response**	**Within a session** (sent as SPDM secured message)	**Outside of a session** (sent as SPDM clear message)	**Session phases**
GET_VERSION	VERSION	Prohibited	Allowed	Not Applicable
GET_CAPABILITIES	CAPABILITIES	Prohibited	Allowed	Not Applicable
NEGOTIATE_ALGORITHMS	ALGORITHMS	Prohibited	Allowed	Not Applicable
GET_PUB_KEY	GET_PUB_KEY_RSP	Prohibited	Allowed	Not Applicable
KEY_EXCHANGE	KEY_EXCHANGE_RSP	Prohibited	Allowed	Not Applicable
GIVE_PUB_KEY	GIVE_PUB_KEY_RSP	Allowed	Prohibited	Session Handshake
FINISH	FINISH_RSP	Allowed	Prohibited	Session Handshake
TPM_CMD, TPM_CMD_L*	TPM_RSP	Allowed	Prohibited (except in Failure Mode)	Application Phase
END_SESSION	END_SESSION_ACK	Allowed	Prohibited	Application Phase
Not Applicable	ERROR	Allowed	Allowed	All Phases
RESPOND_IF_READY	Response to original Request	Allowed	Allowed	All Phases

Table 12: SPDM request and response messages validity

The TPM_RSP response MAY only be sent as SPDM clear message when a TPM implementation with an SPDM secure session layer is in TPM Failure Mode, i.e., TPM_RSP transports the TPM error response TPM_RC_FAILURE (see section 5.5.3.2.1).

 © TCG 2025

5.5.3.1.2 TPM vs. SPDM Response

A TPM implementation with an SPDM secure session layer SHALL respond with:

- An (unencapsulated) TPM response to an (unencapsulated) TPM command
- An SPDM response to an SPDM request (including an encapsulated TPM command)

5.5.3.2 Special Cases

This section specifies the message flow requirements for special cases such as error or Field Upgrade.

5.5.3.2.1 Error Cases

Table 13 specifies the different error cases, which are illustrated in Figure 7, and defines the output for each error case. (For Failure Mode error cases, see section 5.5.3.2.1.1.)

Error case	Input	Output
Outside of a session		
SPDM request fails	SPDM clear request message	SPDM clear response message: ERROR (e.g. InvalidRequest) (as specified in the SPDM specification [1])
Within a session		
SPDM request fails	SPDM secured request message (except TPM_CMD and TPM_CMD_L*)	SPDM secured response message: ERROR (e.g. InvalidRequest) (as specified in the SPDM specification [1])
(Encapsulated) TPM command fails	SPDM secured request message: TPM_CMD or TPM_CMD_L*, which encapsulates TPM command	SPDM secured response message: TPM_RSP, which encapsulates TPM error response (e.g. TPM_RC_SIZE)

Table 13: Error cases

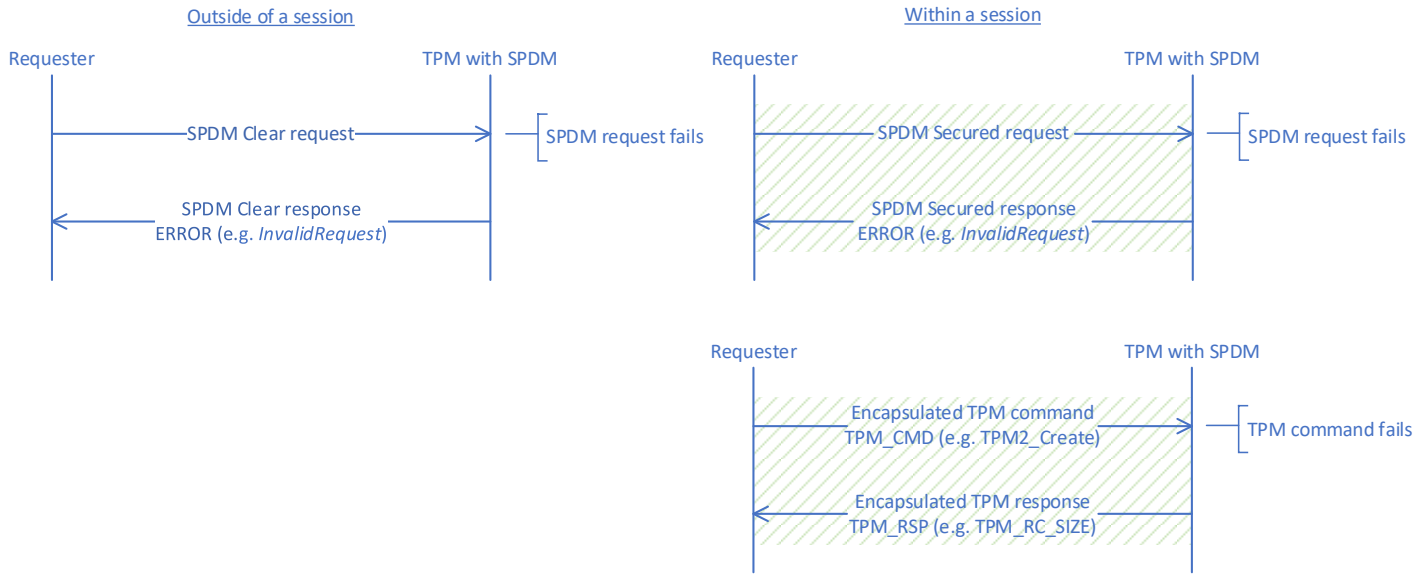


Figure 7: Error cases

5.5.3.2.1.1 Failure Mode

SPDM clear and secured messages MAY cause a TPM implementation with an SPDM secure session layer to enter Failure Mode. A TPM implementation with an SPDM secure session layer SHALL NOT enter Failure Mode unless

 © TCG 2025

the conditions specified in the TPM Library specification [2] occur. These conditions, that are also applicable to SPDM messages, are failed self-tests and failed internal tests. A TPM implementation with an SPDM secure session layer SHALL enter Failure Mode only in the case of a critical/unexpected error resulting from failed self-tests or failed internal tests.

In Failure Mode, a TPM's SPDM secure session layer SHALL terminate all active SPDM connections and sessions.

Start of informative comment

Failure Mode is identical to that defined in the TPM Library specification [2] and the PTP [7]. In Failure Mode, a TPM returns TPM_RC_FAILURE in response to any TPM command except TPM2_GetTestResult or TPM2_GetCapability and a _TPM_Init is required to exit from Failure Mode.

A TPM might enter Failure Mode due to a failed internal test during a cryptographic operation. Cryptographic operations are also performed during SPDM messages such as KEY_EXCHANGE or FINISH. SPDM connections are terminated to prevent cryptographic operations from being performed during Failure Mode.

End of informative comment

Table 14 specifies the different error cases that might cause Failure Mode, which are illustrated in Figure 8, and defines the output for each error case.

Error case	Input	Output
Outside of a session		
SPDM request causes Failure Mode	SPDM clear request message	SPDM clear response message: ERROR (ResetRequired)
Within a session		
SPDM request causes Failure Mode	SPDM secured request message (except TPM_CMD and TPM_CMD_L*)	SPDM clear response message: ERROR (ResetRequired)
(Encapsulated) TPM command causes Failure Mode	SPDM secured request message: TPM_CMD or TPM_CMD_L*, which encapsulates TPM command	SPDM clear response message: TPM_RSP, which transports the TPM error response for TPM_RC_FAILURE (see Figure 9)

Table 14: Failure Mode error cases

While in Failure Mode, a TPM implementation with an SPDM secure session layer SHALL respond with an ERROR message of ErrorCode=ResetRequired if an SPDM clear message (identified via the tag 0x8101) or SPDM secured message (identified via the tag 0x8201) is received.

A TPM's SPDM secure session layer SHALL set *ConnectionHandle* (see section 5.6.3) to zero in any SPDM response message returned in Failure Mode (since no connection information is available in this case).

 © TCG 2025

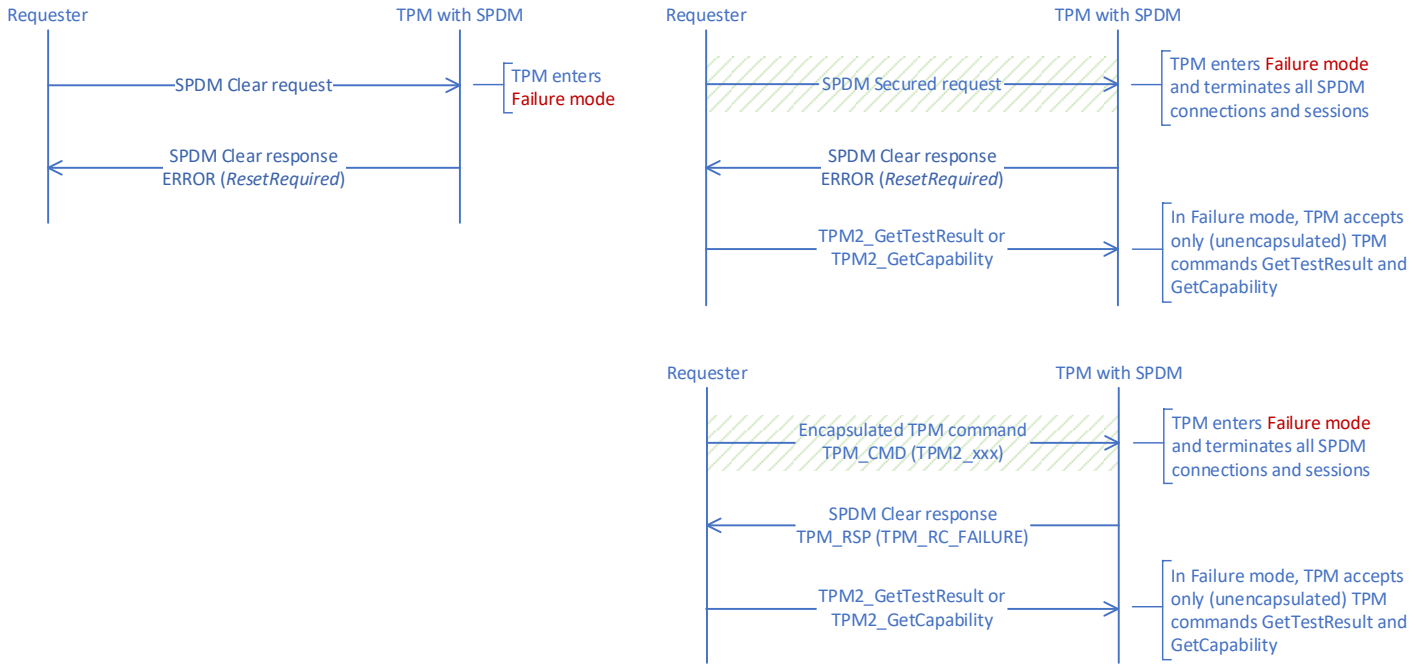


Figure 8: Failure Mode error cases

Figure 9 illustrates how the TPM response for TPM_RC_FAILURE is transported in an TPM_RSP clear message.

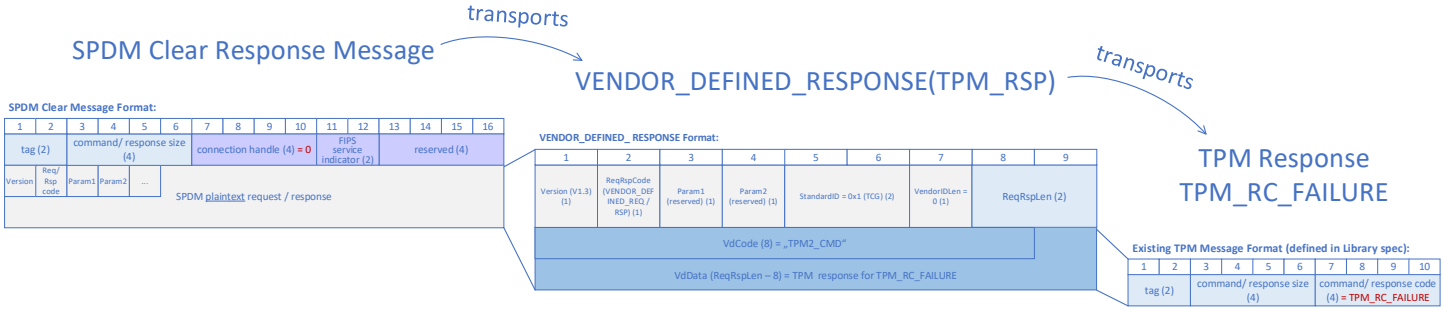


Figure 9: SPDM clear response message TPM_RSP

Start of informative comment

Note that a TPM_CMD (or TPM_CMD_L*) request message can cause a TPM implementation with an SPDM secure session layer to enter Failure Mode but once the TPM is in Failure Mode, it cannot determine that an SPDM request is a TPM_CMD (or TPM_CMD_L*) request message because it cannot unwrap the secured message (since all SPDM sessions were terminated when the TPM entered Failure Mode).

End of informative comment

A TPM implementation with an SPDM secure session layer SHALL make Failure Mode information available in its SPDM layer and application layer via internal protected capabilities as defined in the TPM Library specification [2]. That means:

- If Failure Mode is triggered by a TPM's SPDM secure session layer, the Failure Mode information SHALL be passed to the TPM's application layer.
- If Failure Mode is triggered by a TPM's application layer, the Failure Mode information SHALL be passed to the TPM's SPDM secure session layer.

 © TCG 2025

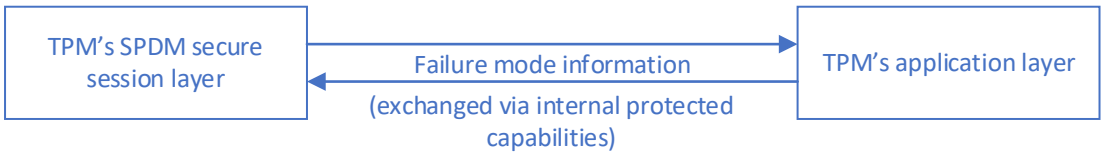


Figure 10: Exchange of Failure Mode information

Start of informative comment

Failure Mode in a TPM's SPDM secure session layer may be caused by an SPDM clear or secured message due to a failure during SPDM request processing or during wrapping/unwrapping of a SPDM secured message (using authenticated encryption).

Failure Mode in a TPM's application layer may be caused by an encapsulated or unencapsulated TPM command due to a failure during TPM command processing.

SPDM connections and sessions are terminated (by a TPM's SPDM secure session layer) regardless of which layer triggered the Failure Mode.

End of informative comment

5.5.3.2.2 Field Upgrade

Whether a TPM implementation with an SPDM secure session layer accepts

- TPM Field Upgrade commands within an SPDM session, or
- SPDM messages during Field Upgrade

is vendor-specific (see section 8).

If a TPM implementation with an SPDM secure session layer does not accept Field Upgrade commands within an SPDM session, it SHALL fail with an ERROR message of ErrorCode=Vd_UnsupportedTpmCmd if it receives an encapsulated Field Upgrade start command.

If a TPM implementation with an SPDM secure session layer does not accept SPDM messages during Field Upgrade, it SHALL fail with the (unencapsulated) TPM error response TPM_RC_BAD_TAG (this is the same error as returned by a TPM implementation without an SPDM secure session layer.)

5.6 SPDM Payload Format

This section specifies the SPDM payload format. There are two types of SPDM payloads (which are differentiated via a tag):

- SPDM clear message – transports unencrypted SPDM messages
- SPDM secured message – transports encrypted and integrity-protected SPDM messages

Mimicking the TPM command structure defined in the TPM Library specification [2], each SPDM payload SHALL have a header that consists of:

- a 2-byte tag
- a 4-byte (command/ response) size

Start of informative comment

Prepending the tag and size ensures compatibility with the TPM command structure and avoids that existing TPM SPI/ I2C drivers and protocol implementations have to be changed. To ensure compatibility, the size is required to be at the same location, i.e., byte 3-6.

End of informative comment

 © TCG 2025

As illustrated in Figure 11, the same tag-size structure is used for:

- a TPM message (defined in the TPM Library specification [2])
- an SPDM clear message
- an SPDM secured message

The fields depicted in Figure 11 and Figure 12 are specified in Table 15 and Table 20.

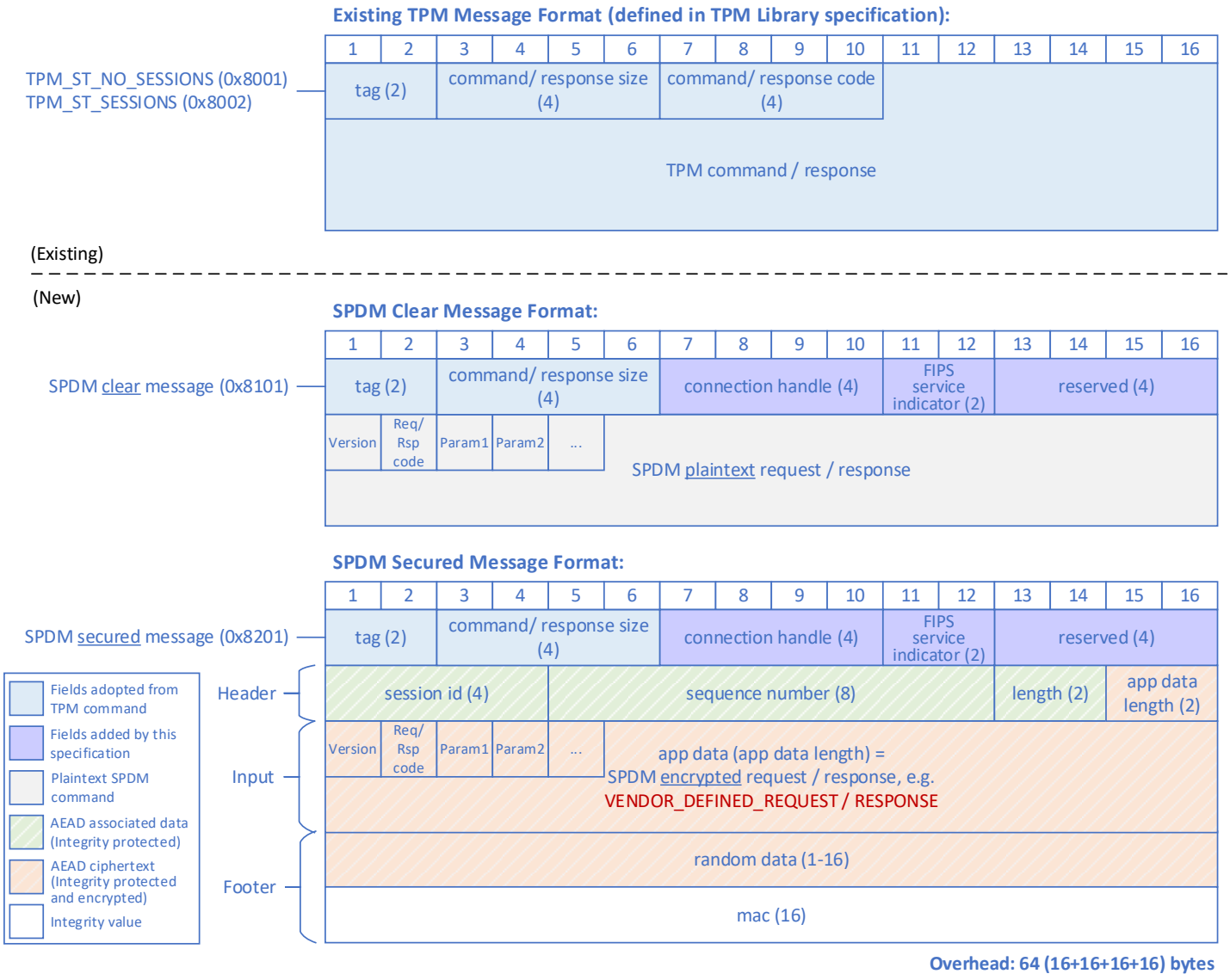


Figure 11: SPDM payload format

 © TCG 2025

Figure 12 illustrates a VENDOR_DEFINED_REQUEST or VENDOR_DEFINED_RESPONSE as specified in [1].

Start of informative comment

Note that the “VendorID” field is absent because “VendorIDLen” is zero.

End of informative comment

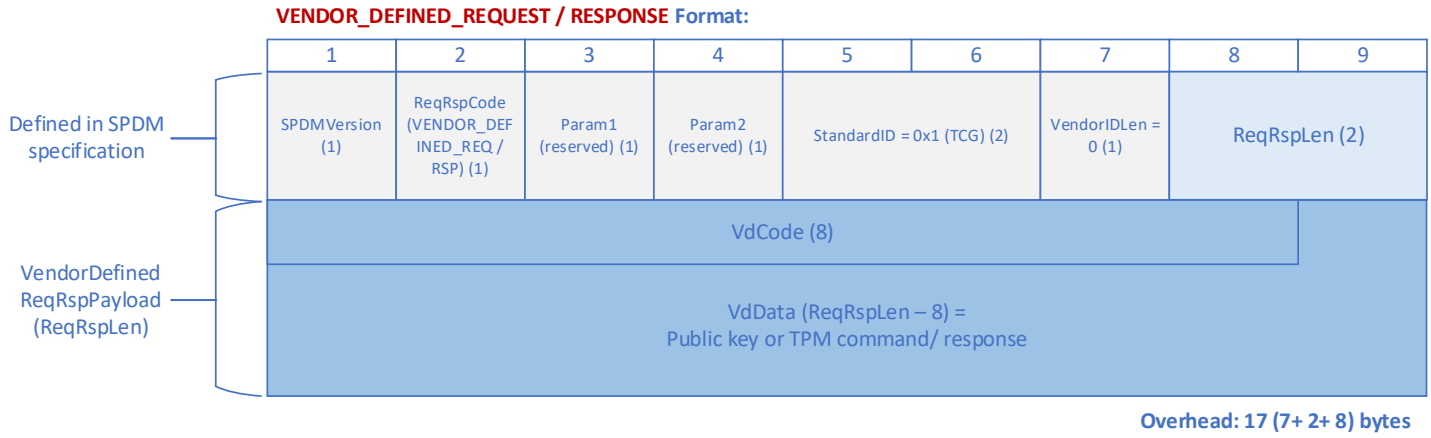


Figure 12: Vendor-defined request/ response

Figure 13 illustrates a TPM command that is encapsulated in a VENDOR_DEFINED_REQUEST (TPM_CMD), which in turn is encapsulated in an SPDM secured message.

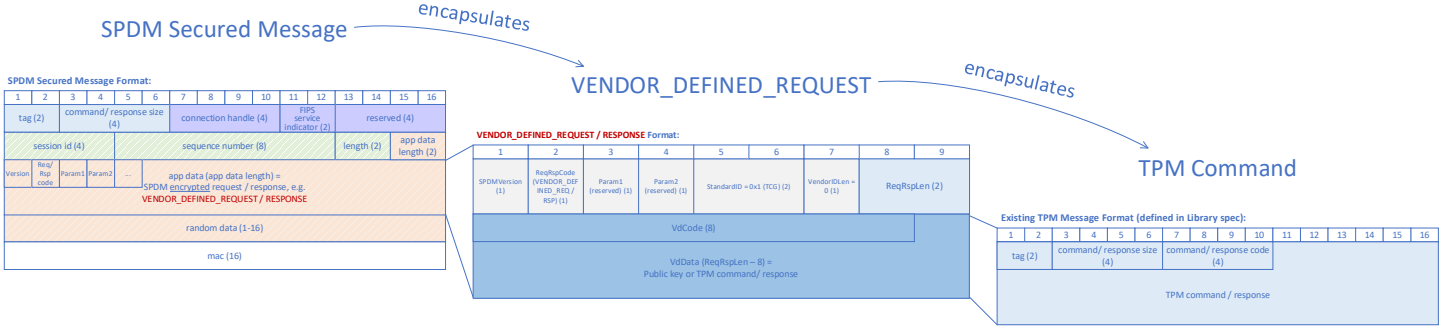


Figure 13: TPM command encapsulated in SPDM secured message

5.6.1 SPDM Payload Byte Ordering

This section specifies the endianness for the different SPDM payload fields (which are depicted in Figure 11).

The fields in Table 15 that are defined in the SPDM specifications [1] and [6] (highlighted in blue) SHALL follow the endianness defined by the SPDM specifications (which is little endian format).

All other fields in Table 15 SHALL follow the endianness of the TPM defined in the TPM Library specification [2] (which is big endian format).

 © TCG 2025

Byte Offset c=clear, s=secured	Fields	Type	Byte Ordering	Description
	Header			
0	SPDM message tag	uint16	TPM convention (big endian)	Tag for SPDM clear or secured message, see Table 16
2	Command / response size	uint32	TPM convention (big endian)	Command or response size
6	Connection handle	uint32	TPM convention (big endian)	Identifier for SPDM connection, see section 5.6.3
10	FIPS service indicator	uint16	TPM convention (big endian)	FIPS service indicator for SPDM request and TPM command, see section 5.6.4
12	Reserved			Reserved for future use
	SPDM message			
16 (s)	SPDM secured message fields	byte[]	SPDM convention (little endian - except for the MAC, Application Data and Random Data)	Defined in DSP0277, “Secured Messages using SPDM” specification [6], see Table 20
16 (c)/ 32 (s)	SPDM request / response message	byte[]	SPDM convention (little endian)	Defined in SPDM specification [1]
	VendorDefinedReqRspPayload			
25 (c)/ 41 (s)	VdCode	uint64	TPM convention (big endian)	Code for TCG specified vendor-defined commands, see Table 11
33 (c)/ 49 (s)	VdData	byte[]	TPM convention (big endian)	Data for TCG specified vendor-defined commands (either public key or TPM command / response), see Table 11

Table 15: SPDM message field byte ordering

5.6.2 SPDM Payload Tags

Table 16 specifies the SPDM payload tags for an SPDM clear and secured message.

SPDM Payload Tag	Value (big endian)
SPDM clear message tag	0x8101
SPDM secured message tag	0x8201

Table 16: SPDM message tags

For an invalid tag, a TPM implementation with an SPDM secure session layer SHALL return the TPM’s response for a bad tag (TPM_RC_BAD_TAG) as specified in the TPM Library specification [2].

 © TCG 2025

That means, a host's SPDM secure session layer (that implement this TPM SPDM secure transport protocol) is required to be able to process a TPM error response with a *responseCode* of TPM_RC_BAD_TAG.

Start of informative comment

In the case of an invalid tag, a TPM implementation with an SPDM secure session layer is not able to determine whether the host intended to send a TPM command or an SPDM request (and therefore decide whether to return an TPM or SPDM error response). For consistency, a TPM with an SPDM secure session layer always returns the TPM error response TPM_RC_BAD_TAG in this case.

Note that the application data is confidentiality and integrity protected in an SPDM secured message. This TPM SPDM secure transport protocol does not support an option for application data to be integrity protected only.

There is no dependency between the SPDM message tag defined in Table 16 and the TPM command/ response tag (TPM_ST_NO_SESSIONS or TPM_ST_SESSIONS) defined in the TPM Library specification [2]. TPM_CMD (or TPM_CMD_L*) and TPM_RSP sent as an SPDM secured message (tagged with 0x8201) can transport a TPM command/ response (see Figure 13), which might or might not have a TPM authorization session (tagged with 0x8001 or 0x8002).

End of informative comment

5.6.3 Connection Handle

The *ConnectionHandle* identifies the SPDM connection. The handle is 4-bytes and is included in the SPDM (clear and secured) message header (see Figure 11). For the setting of *ConnectionHandle*, see section 5.7.1.

5.6.4 FIPS Service Indicator

The FIPS service indicator is 2-bytes and is included in the SPDM (clear and secured) message header (see Figure 11). The FIPS service indicator SHALL be set by the TPM's SPDM secure session layer in the SPDM response message and ignored in the SPDM request message.

The FIPS service indicator indicates whether

1) the SPDM request is a FIPS-approved service, and
2) in addition, for the TPM_CMD (or TPM_CMD_L*) request, which encapsulates a TPM command, whether the TPM command is a FIPS-approved service.

The FIPS service indicator is defined in Table 17 and is divided into two sub-indicators, to separately indicate whether the SPDM request message and the encapsulated TPM command are FIPS-approved services.

Bit	(Sub-)Indicator	Description
1:0	FIPS_140_3_INDICATOR _SPDM_REQ	SPDM request FIPS service indicator: Indicates whether the SPDM request is a FIPS-approved service. If the SPDM request encapsulates a TPM command, these bits do not indicate whether the TPM command is a FIPS-approved service.
7:2	Reserved	Set to zero
9:8	FIPS_140_3_INDICATOR _TPM_CMD	TPM command FIPS service indicator: Indicates whether the TPM command that is encapsulated in the SPDM request is a FIPS-approved service. These bits SHALL be a copy of the FIPS 140-3 service indicator bits, which are reported by the TPM in TPMA_MODES with TPM2_GetCapability in bit 3:2 (FIPS_140_3_INDICATOR). These bits SHALL only be set for the TPM_CMD (or TPM_CMD_L*) request message and SHALL be zero for all other SPDM requests.

 © TCG 2025

15:10	Reserved	Set to zero

Table 17: FIPS 140-3 Service Indicator

The SPDM request FIPS service indicator SHALL use the same status values as the TPM command FIPS service indicator.

Start of informative comment

The TPM Library specification [2] defines the following status values for the FIPS service indicator:

(00): service belongs to category 2, 3 or 5 (non-security relevant services)

(01): service belongs to category 1 (approved security services)

(10): service belongs to category 4 (non-approved security services)

(11): reserved value

NOTE The categories 1 to 5 are defined in FIPS 140-3 IG, section 2.4.C [9]

How the TPM command FIPS service indicator should be set for each TPM command is described in the TCG FIPS 140-3 Guidance [10].

Table 18 provides guidance on how the SPDM request FIPS service indicator should be set for each SPDM request. Vendors should consult the relevant FIPS standards and the FIPS 140-3 Implementation Guide (IG) [9]. (The cells highlighted in lighter grey correspond to the lighter grey cells in Table 19.)

	SPDM Request	**Approved status (of SPDM Request)**	**SPDM Request FIPS service indicator** (FIPS_140_3_INDICATOR _SPDM_REQ)	**TPM Command FIPS service indicator** (FIPS_140_3_INDICATOR _TPM_CMD)
Clear messages	GET_VERSION	Non-security relevant services or Approved security services (see Note 1)	00 or 01	n/a (00)
	GET_CAPABILITIES		00 or 01	n/a (00)
	NEGOTIATE_ALGORITHMS		00 or 01	n/a (00)
	GET_PUB_KEY		00 or 01	n/a (00)
	KEY_EXCHANGE	Approved security services (see Note 1)	01	n/a (00)
Secured messages	GIVE_PUB_KEY		01	n/a (00)
	FINISH		01	n/a (00)
	TPM_CMD, TPM_CMD_L* (SPDM request that encapsulates TPM command)		01	Copy of TPMA_MODES, bit 3:2 (FIPS_140_3_INDICATOR) (see Note 2)
	END_SESSION		01	n/a (00)

Table 18: Approved status of SPDM requests

Note 1:

To be a FIPS-approved service, a SPDM request has to use FIPS-approved algorithms.

This specification requires:

- ECDSA and ECDH (with NIST curves)
- SHA-256 or above

 © TCG 2025

- AES with GCM

and allows additional algorithms that are supported by the SPDM specification (see section 5.5.1.3.1).

Whether the (non-cryptographic) SPDM clear messages are approved services or non-security relevant, is implementation-specific because it depends on when the transcript hash is computed.

For FIPS compliance, each algorithm requires a cryptographic algorithm self-test (Known Answer Test (KAT)) before the algorithm is used the first time after power-up. At time of writing, GCM is not supported by TPMs: therefore, along with the algorithm, a TPM implementation with an SPDM secure session layer will need to include a KAT for GCM, in order to be FIPS compliant.

If FIPS self-tests are performed during an SPDM message, then the self-tests should be performed during an SPDM request message that performs cryptographic operations in order to not violate the timing requirement for non-cryptographic request messages specified in the SPDM specification [1] (“Table 7 — Timing specification for SPDM messages", see parameter “ST1”).

Note 2:

The reason for copying TPMA_MODES, bits 3:2 (FIPS_140_3_INDICATOR) in the SPDM message header is to make it more convenient for calling code to retrieve the FIPS service indicator, as this avoids having to read it with TPM2_GetCapability after each TPM command.

The overall Approved status of the TPM_CMD (or TPM_CMD_L*) request with its encapsulated TPM command is obtained by combining the two service indicators as detailed in Table 19. (The cells highlighted in lighter grey correspond to the lighter grey cells in Table 18.)

SPDM Request FIPS service indicator for TPM_CMD, TPM_CMD_L* (FIPS_140_3_INDICATOR _SPDM_REQ)	**TPM Command FIPS service indicator** (FIPS_140_3_INDICATOR _TPM_CMD)	**Overall Approved status**
01 (approved)	00 (non-security relevant)	01 (approved)
01 (approved)	01 (approved)	01 (approved)
01 (approved)	10 (non-approved)	10 (non-approved)

Table 19: Overall Approved status of TPM_CMD request

For all other SPDM requests, the overall Approved status is equal to the status reported in the SPDM Request FIPS service indicator.

End of informative comment

The FIPS service indicator SHALL only be set in the SPDM response if the request is executed successfully. The TPM’s SPDM secure session layer SHALL set the FIPS service indicator to zero (0x0000) in an ERROR response message.

The FIPS service indicator is required for FIPS 140-3 validation.

- The SPDM secure session layer of a TPM that does not comply with FIPS 140-3 MAY set the FIPS service indicator to zero (0x0000) for all SPDM response messages.
- The SPDM secure session layer of a TPM that complies with FIPS 140-3 SHOULD NOT support FIPS non-approved algorithms.

5.6.5 SPDM Secured Message Format

The secured message format SHALL follow the format specified by DMTF in DSP0277, “Secured Messages using SPDM Specification” [6]. Table 20 specifies the fields of an SPDM secured message used in this TPM SPDM

 © TCG 2025

secure transport protocol and is derived from Table “Secured Message fields definition” specified in [6]. The variable-length fields for which the length is defined by this binding specification are highlighted in blue.

A TPM’s SPDM secure session layer SHALL implement the fields and length as defined in Table 20.

Note that [6] specifies that all fields are little endian, except for the MAC and Application Data.

Offset	Field	Length (bytes)	Description
0	Session ID	4	Session identifier: *ReqSessionID* || *RspSessionID* (exchanged during KEY_EXCHANGE)
4	Sequence Number	8	A counter that is incremented after each message encryption, XOR’ed with the IV. Request and response messages use different dedicated counters Note: Used for error handling only. SPDM Requester and Responder maintain their own message counters, and do not rely upon the value of Sequence Number in messages.
12	Length	2	Length of all following data in the SPDM secured message, including the MAC
14	Application Data Length	2	Length of Application Data
16	Application Data	P (Application Data Length)	SPDM request/ response
16 + P	Random Data	R (1-16)	Random data of random length (between 1-16 bytes) to disguise the real length of Application Data Note: The length of Random Data is determined by calculating: Length – 2 – Application Data Length – 16 (length of MAC)
16 + P + R	Message Authentication Code (MAC)	16	AEAD tag

Table 20: Secured message fields

5.7 SPDM Connections, Sessions

This TPM SPDM secure transport protocol is designed to support multiple SPDM connections, where each connection supports at least one SPDM session (see section 5.7.2).

A TPM’s SPDM secure session layer SHALL:

- Support at least one SPDM connection and one SPDM session
- Support at least the same number of SPDM sessions as SPDM connections
- If multiple connections are supported, support all sessions established either within the same connection or different connections

Start of informative comment

 © TCG 2025

For example, if a TPM's SPDM secure session layer supports in total two connections and five sessions, then it must be possible for a Requester to establish either all five sessions within a single connection or establish the five sessions distributed over two connections.

End of informative comment

5.7.1 SPDM Connection

A connection is identified via a 4-byte *ConnectionHandle* (see section 5.6.3), which is transmitted as part of the SPDM message header (see Figure 11).

In the first request that starts a connection (i.e., GET_VERSION or - if CACHE_CAP is supported - GET_PUB_KEY), the Requester sets *ConnectionHandle* to zero. A TPM's SPDM secure session layer creates an SPDM context for the connection and assigns a handle: the handle is returned in *ConnectionHandle* in the response. If the TPM's SPDM secure session layer supports only one connection, the handle is set to zero (since no differentiation between different connections is needed).

The same *ConnectionHandle* is used by the Requester in subsequent requests that belong to the same connection. The TPM's SPDM secure session layer echoes the *ConnectionHandle* in the response.

When a connection is terminated, the associated *ConnectionHandle* is released and MAY be assigned by the TPM's SPDM secure session layer for a new connection.

A connection can be reset using GET_VERSION with *ConnectionHandle* set to the handle of an existing connection (which is to be reset). When reset, the *ConnectionHandle* remains unchanged.

If a TPM's SPDM secure session layer supports only one connection, it SHALL:

- If GET_VERSION uses a value of *ConnectionHandle* that is equal to zero,
  - If no connection exists, create a connection and set *ConnectionHandle* in the response to zero
  - Otherwise, reset the connection and terminate the active sessions
- If GET_VERSION uses a value of *ConnectionHandle* that is not equal to zero,
  - Return an ERROR message of ErrorCode=Vd_InvalidConnHandle

If a TPM's SPDM secure session layer supports multiple connections, it SHALL:

- If GET_VERSION uses a value of *ConnectionHandle* that is equal to zero,
  - If there are available connections, create a connection and set *ConnectionHandle* in the response to the assigned handle (which is unequal to zero)
  - Otherwise, return an ERROR message of ErrorCode=Vd_ConnLimitExceeded
- If GET_VERSION uses a value of *ConnectionHandle* that is not equal to zero,
  - If *ConnectionHandle* is associated with an active connection, reset that connection and terminate the active sessions associated with that connection
  - Otherwise, return an ERROR message of ErrorCode=Vd_InvalidConnHandle

The assigned *ConnectionHandle* value SHALL uniquely identify the connection. If only one connection is supported, the *ConnectionHandle* SHALL be 0.

If multiple connections are supported, *ConnectionHandle* SHALL be a counter, starting at 1, up to the maximum number of supported connections. This allows a Requester to reset connections if the Requester loses its connection handle.

Start of informative comment

 © TCG 2025

ConnectionHandles may be mapped, for example, to localities or processes. E.g., each locality may be assigned one *ConnectionHandle* or one locality may be assigned multiple *ConnectionHandles*, where each process running in the same locality is assigned a different *ConnectionHandle*. This allows e.g., one process to share a session with another process by handing over its SPDM session context (including *ConnectionHandle* and session keys).

End of informative comment

5.7.2 SPDM Session

Start of informative comment

An SPDM session is started with KEY_EXCHANGE and is identified via a 4-byte session ID (see Table 20).
The algorithms for the SPDM session are negotiated per connection. Therefore, all sessions within the same connection use the same algorithms. Sessions within different connections can use different algorithms.
All sessions within the same connection use the same signature verification type, i.e., either mutual or Responder-only signature verification (since it is determined during GET_CAPABILITIES).

End of informative comment

Each session can use a different Requester public key since the Requester public key is provided to a TPM's SPDM secure session layer in GIVE_PUB_KEY following KEY_EXCHANGE. The Responder public key is read only once per connection (since GET_PUB_KEY is executed before KEY_EXCHANGE).

Figure 14 shows multiple connection and sessions, where the TPM and Requester support multiple algorithms and SPDM-identity keys. This specification requires a TPM's SPDM secure session layer to support only one connection and session and one set of algorithms (see section 5.5.1.3.1).

 © TCG 2025

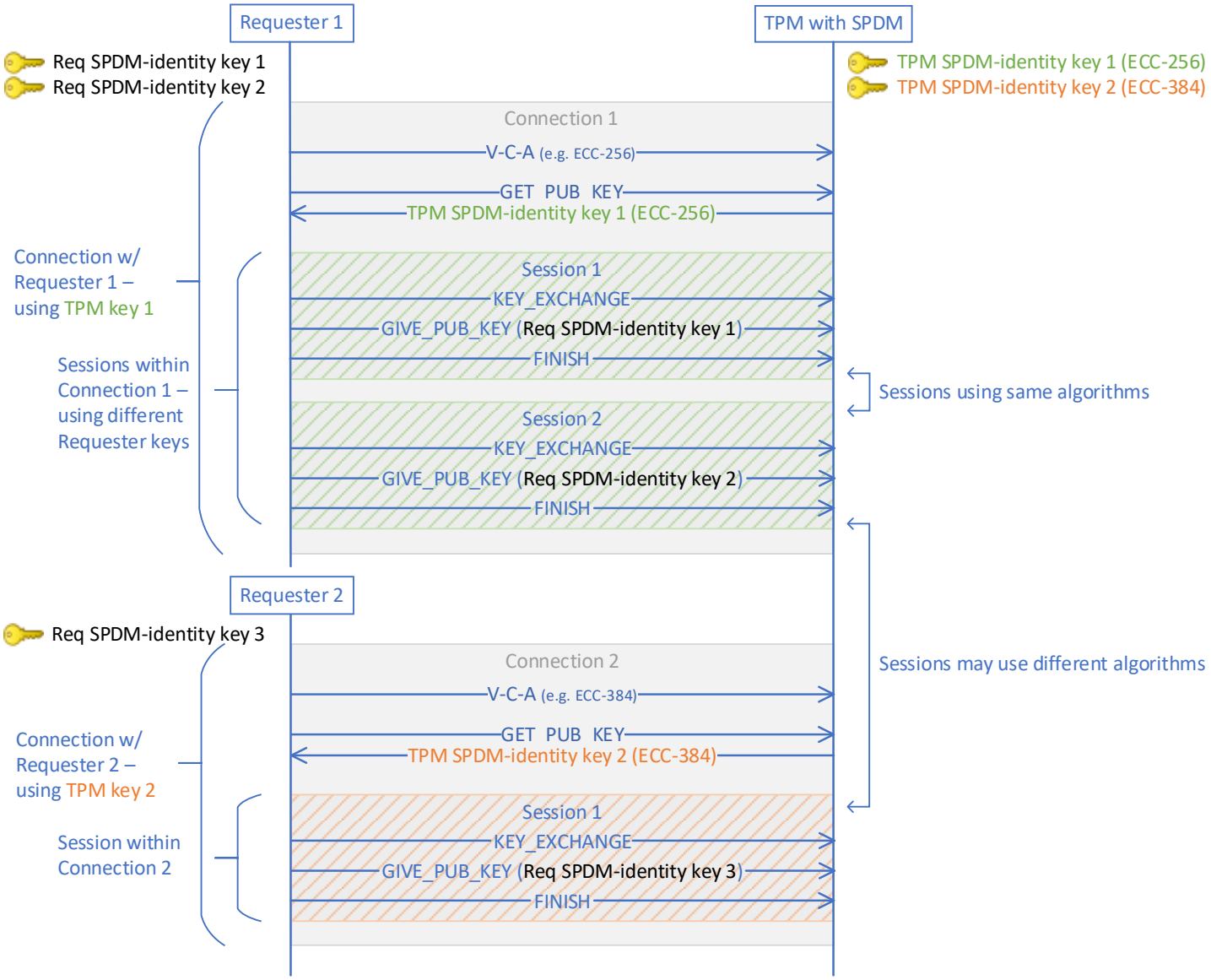


Figure 14: SPDM connections and sessions

Figure 15 depicts SPDM sessions started within different connections where one of the connections is reset with GET_VERSION.

 © TCG 2025

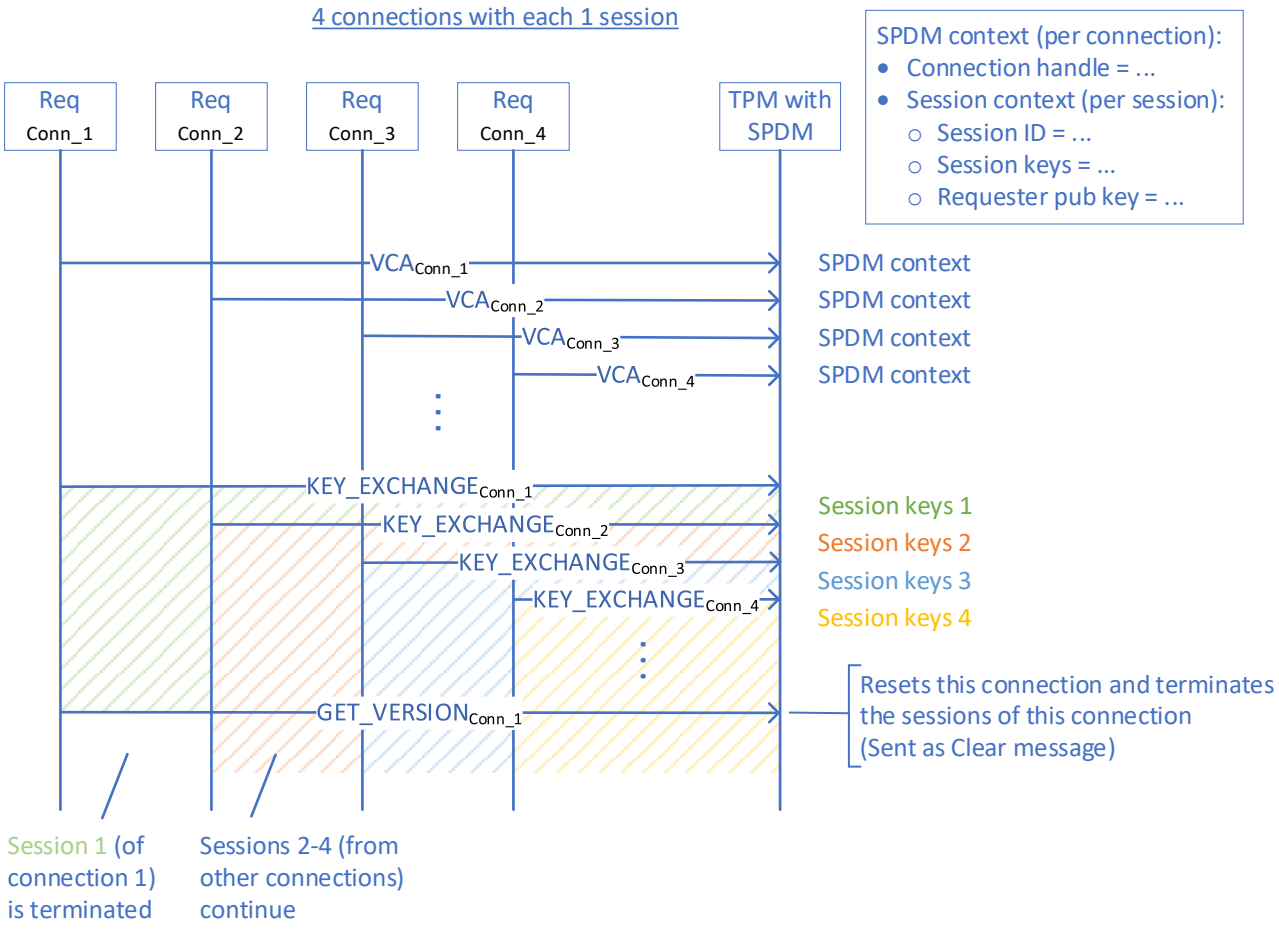


Figure 15: Example with multiple connections

5.7.2.1 Special TPM Commands and Signals

5.7.2.1.1 TPM_Hash_Start/ Data / End Signals

SPDM sessions are used to protect TPM commands but not the _TPM_Hash_Start, _TPM_Hash_Data, TPM_Hash_End signals, which are out of scope of this specification.

Start of informative comment

_TPM_Hash_Start, _TPM_Hash_Data, TPM_Hash_End signals are sent from Locality 4. Locality 4 accesses are controlled by trusted hardware responsible for maintaining RTM flows (such as D-RTM and H-CRTM), and not used by software to send TPM commands.

End of informative comment

5.7.2.1.2 Startup Sequence

Figure 16 illustrates the TPM startup sequence, where TPM2_Startup is sent outside (figure left) and within an SPDM session (figure right). The diagrams also depict the _TPM_Hash_Start/ Data/ End signals (sent either before or after TPM2_Startup).

Start of informative comment

Some Platform TPM Profiles (such as the PTP [7]) specify startup timing requirements after which a TPM must be ready to receive unencapsulated TPM commands after _TPM_Init (i.e., TPM2_Startup).

 © TCG 2025

If TPM2_Startup is encapsulated in an SPDM message, platforms should expect the TPM startup phase to be longer because SPDM messages have to be exchanged to establish an SPDM session.

A TPM implementation with an SPDM secure session layer should be ready to receive SPDM request messages as soon as it is ready to receive unencapsulated TPM commands.

End of informative comment

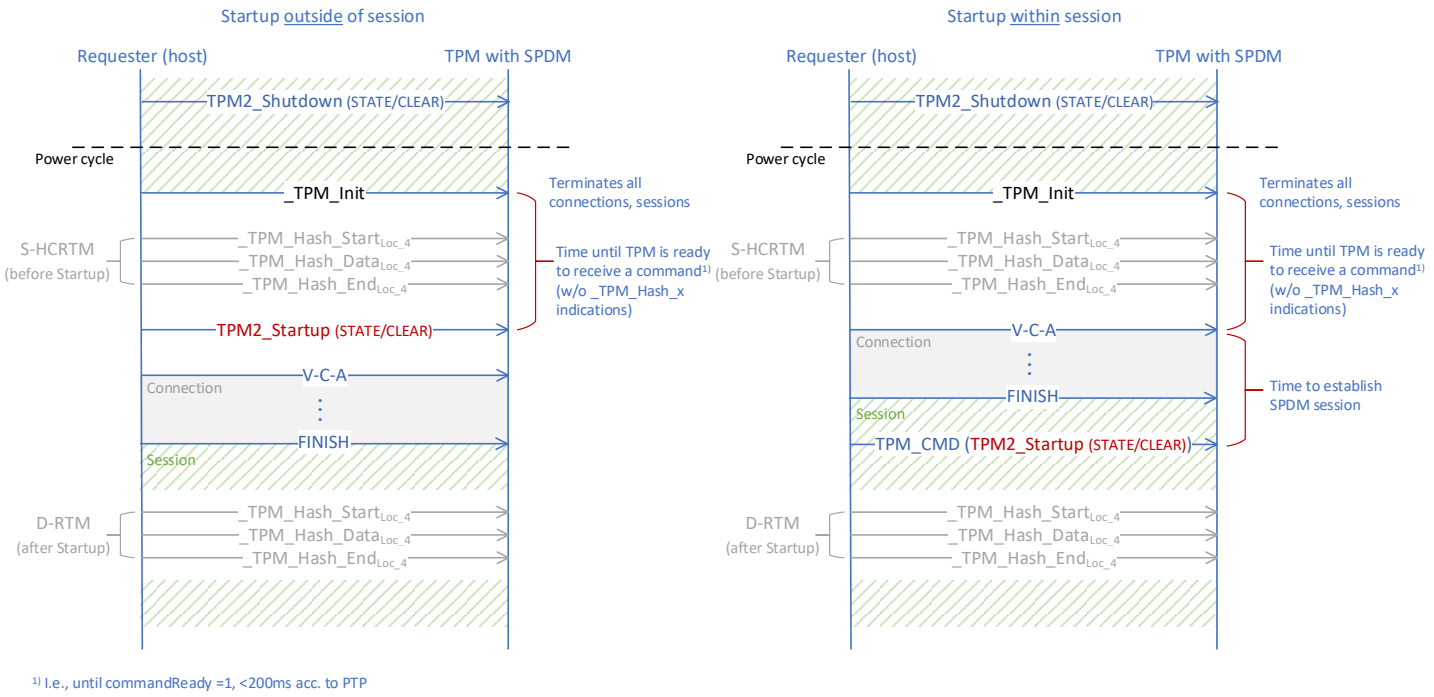


Figure 16: Startup sequence

5.7.2.2 TPM Command Cancel and Abort

Start of informative comment

The PC Client Platform TPM Profile (PTP) [7] specifies a Command Cancel and Abort mechanism, which allow a host to request a TPM to terminate processing the current TPM command. On a successful Command Cancel, a TPM returns TPM_RC_CANCELLED, while on a successful Command Abort, a TPM returns no response.

The Command Cancel and Abort requirements specified in [7] for (unencapsulated) TPM commands apply also to encapsulated TPM commands.

End of informative comment

An active SPDM session SHALL NOT be terminated by a Command Cancel or Abort operation.

5.7.2.2.1 Command Cancel

On a Command Cancel request, a TPM with an SPDM secure session layer SHALL NOT cancel processing that happens in the SPDM secure session layer (i.e., an SPDM layer has no cancel checkpoints).

SPDM messages (with the exception of TPM_CMD and TPM_CMD_L*) SHALL NOT be cancelable (via the TPM Command Cancel mechanism). TPM_CMD (and TPM_CMD_L*) SHALL be cancelable only during the TPM command execution, but not during the processing of the SPDM secured message (i.e., during the wrapping/unwrapping of the request/response message).

 © TCG 2025

When an encapsulated TPM command is (successfully) canceled, a TPM with an SPDM secure session layer SHALL return TPM_RC_CANCELLED in an encapsulated TPM response. That means, the Requester and Responder session sequence numbers are incremented as usual.

5.7.2.2.2 Command Abort

Start of informative comment

The purpose of Command Abort is to resynchronize the communication with a TPM. Instead of a physical abort on the interface, it is recommended that a host resynchronizes on SPDM protocol level by sending GET_VERSION.

End of informative comment

On a Command Abort request, a TPM with an SPDM secure session layer MAY abort processing that happens in the SPDM secure session layer (i.e., an SPDM layer can have abort checkpoints).

SPDM messages MAY be abortable (via the TPM Command Abort mechanism). As required by the PTP [7], TPM_CMD (and TPM_CMD_L*) must be abortable during the TPM command execution. TPM_CMD (and TPM_CMD_L*) MAY be abortable during the processing of the SPDM secured message (i.e., during the wrapping/unwrapping of the request/response message).

When an SPDM message or encapsulated TPM command is (successfully) aborted, a TPM with an SPDM secure session layer:

- SHALL NOT increment the Responder session sequence number and IV (since no response is returned)
- SHALL increment the Requester session sequence number and IV (since the secured message request was processed)

5.7.2.3 Enforcing SPDM Session Protection via TPM Policy

For a TPM object to be protected by an SPDM secure session that is compliant with this specification, authorization of the TPM object is required to require policy and its *authPolicy* is required to include TPM2_PolicyTransportSPDM.

A TPM's SPDM secure session layer SHALL pass the following secure channel information to a TPM's application layer via internal protected capabilities as defined in the TPM Library specification [2]:

- The information whether a TPM command is sent within an SPDM session, i.e., whether it is encapsulated in an SPDM secured message
- The corresponding Requester and TPM SPDM-identity public keys used in the session negotiation

Start of informative comment

The TPM Library specification [2] defines a TPM2_PolicyTransportSPDM command for use with the protocol defined in this specification. This policy command causes conditional gating of a policy based on the presence of a secure transport channel.

To protect sensitive data or keys from being accessed outside of an SPDM session, all such sensitive data or keys require an authorization policy (*authPolicy*) that includes TPM2_PolicyTransportSPDM. TPM2_PolicyTransportSPDM allows the policy to be bound to a specific Requester SPDM-identity key and/or TPM SPDM-identity key (via their Names). To enforce mutual signature verification, a Requester key Name has to be provided (in the *reqKeyName* parameter). TPM2_PolicyTransportSPDM is a deferred assertion, i.e., the presence of an SPDM session and the SPDM-identity key Names are verified when the TPM policy session is used for authorization.

End of informative comment

 © TCG 2025

5.7.3 Terminating a Connection, Session

An SPDM connection or session SHALL be terminated via the SPDM Request or event described in Table 21. When an SPDM session is terminated, a TPM's SPDM secure session layer SHALL zeroize all SPDM session secrets.

SPDM Request / Event	Effect on SPDM Connections	Effect on SPDM Sessions
GET_VERSION	Resets connection associated with connection handle	Terminates all sessions of the connection associated with connection handle
END_SESSION	No effect	Terminates specific session associated with session ID
ERROR message of ErrorCode=DecryptError	No effect	Terminates specific session associated with session ID (as specified in [1])
ERROR message of ErrorCode=RequestResynch	Requires the specific connection associated with connection handle to be reset. I.e., the RequestResynch error code is returned until the Requester issues a GET_VERSION (as specified in [1])	Terminates all sessions of the connection associated with connection handle (either directly as result of this error code or indirectly via GET_VERSION)
_TPM_Init	Terminates all connections	Terminates all sessions
Failure Mode	Terminates all connections	Terminates all sessions

Table 21: Command/ events to terminate connection or session

A TPM's SPDM secure session layer SHALL terminate all SPDM connections and sessions on _TPM_Init or when the TPM enters Failure Mode (see section 5.5.3.2.1.1), and preserve them otherwise. That means, SPDM connections and sessions are terminated independent of TPM startup types (i.e. TPM Reset, TPM Restart and TPM Resume).

Start of informative comment

For errors that require resynchronization, the SPDM specification [1] requires a Responder to return an ERROR message of ErrorCode=RequestResynch until the Requester issues a GET_VERSION. The RequestResych error code does not terminate the connection directly because the *ConnectionHandle* is needed to identify subsequent request messages that are associated with the connection (that requires resynchronization).

Terminating SPDM connections and sessions independently of TPM startup types allows a clearer separation of a TPM's SPDM secure session layer from a TPM's application layer. (Until a TPM receives TPM2_Startup(STATE or CLEAR), it cannot determine whether the startup type is TPM Restart or Resume.)

End of informative comment

 © TCG 2025

6 SPDM-IDENTITY KEYS

The two asymmetric signing keys used for signature generation and verification in the handshake of the SPDM protocol are:

- The TPM SPDM-identity key – Used by the TPM's SPDM secure session layer to generate the signature in the KEY_EXCHANGE_RSP response
- The Requester SPDM-identity key – Used by the Requester to generate the signature in the FINISH request (for mutual signature verification only)

Each SPDM Requester can have its own identity and, therefore, signing key. On the TPM side, there is only one TPM SPDM-identity key per supported signature generation algorithm.

Start of informative comment

This specification does not define a mechanism to select a TPM SPDM-identity key. The TPM SPDM-identity key is implicitly selected via the negotiated algorithms.

End of informative comment

The public keys of the SPDM-identity keys – marshalled as TPMT_PUBLIC structures – are transmitted in the GET_PUB_KEY_RSP response and GIVE_PUB_KEY request

6.1 Object Public Structure

This section defines the TPMT_PUBLIC structure of the SPDM-identity keys, including the object attributes.

For the TPM SPDM-identity key, a TPM implementation with an SPDM secure session layer SHALL use the TPMT_PUBLIC structure to:

- Transmit the TPM SPDM-identity public key in the GET_PUB_KEY_RSP response
- Return the TPM SPDM-identity public key in TPM2_GetCapability(capability = TPM_CAP_PUB_KEYS)
- Compute the TPM SPDM-identity key's Name, which is:
  - Used in TPM2_PolicyTransportSPDM (the Name is checked against the provided *tpmKeyName* when the policy is used for authorization)
  - Returned in TPM2_GetCapability(capability = TPM_CAP_SPDM_SESSION_INFO)

For the Requester SPDM-identity key, a TPM implementation with an SPDM secure session layer SHALL use the TPMT_PUBLIC structure as provided in the GIVE_PUB_KEY request to:

- Compute the Requester SPDM-identity key's Name, which is:
  - Used in TPM2_PolicyTransportSPDM (the Name is checked against the provided *reqKeyName* when the policy is used for authorization)
  - Returned in TPM2_GetCapability(capability = TPM_CAP_SPDM_SESSION_INFO)

It is recommended that the Requester sets the TPMT_PUBLIC structure of the Requester SPDM-identity key (transmitted in GIVE_PUB_KEY) as defined in this section.

Start of informative comment

The TPMT_PUBLIC structure normally describes the properties of a TPM object. Since the SPDM-identity keys are not created or used as TPM objects by the TPM application layer, not all parameters have a meaning, since they are not enforced by the TPM. Nonetheless, the TPMT_PUBLIC structure is reused to report the algorithm and basic properties of the key. Object attributes or authorization values that are not applicable to an SPDM-identity key are set to zero.

End of informative comment

 © TCG 2025

6.1.1 Object Attributes (TPMA_OBJECT)

Table 22 defines the object attributes for both a TPM and Requester SPDM-identity key. The TPMA_OBJECT values are summarized in Table 23.

Bit	Name	TPM SPDM key	Requester SPDM key
0	Reserved	0	0
1	fixedTPM	SET (1)	n/a (0)
2	stClear	n/a (0)	n/a (0)
3	Reserved	0	0
4	fixedParent	SET (1)	n/a (0)
5	sensitiveDataOrigin	SET (1)	n/a (0)
6	userWithAuth	n/a (0)	n/a (0)
7	adminWithPolicy	n/a (0)	n/a (0)
8	firmwareLimited	SET (1) or CLEAR (0)	n/a (0)
9	svnLimited	SET (1) or CLEAR (0)	n/a (0)
10	noDA	n/a (0)	n/a (0)
11	encryptedDuplication	n/a (0)	n/a (0)
15:12	Reserved	0	0
16	restricted	SET (1)	CLEAR (0)
17	decrypt	CLEAR (0)	CLEAR (0)
18	sign / encrypt	SET (1)	SET (1)
19	x509sign	n/a (0)	n/a (0)
31:20	Reserved	0	0

Table 22: SPDM-identity key object attributes (TPMA_OBJECT)

SPDM-identity key	TPMA_OBJECT value
TPM SPDM-identity key (neither firmware nor SVN-limited)	0x00050032
TPM SPDM-identity key (firmware-limited)	0x00050132
TPM SPDM-identity key (SVN-limited)	0x00050232
Requester SPDM-identity key	0x00040000

Table 23: TPMA_OBJECT values

6.1.2 Object Public Structures (TPMT_PUBLIC)

A TPM implementation with an SPDM secure session layer that supports:

- Algorithm set B, SHALL use the public structure defined in Table 24 (for ECC NIST P-384)
- Algorithm set C, SHALL use the public structure defined in Table 25 (for ECC NIST P-521)

(For the algorithm set B and C, see section 5.5.1.3.1)

The TPM and Requester SPDM-identity keys MAY only be used with the algorithms (including the scheme hash) defined in their public structures.

Usage of other algorithms (such as RSA or other ECC curves) for the SPDM-identity key are out of scope of this specification.

6.1.2.1 ECC NIST P-384

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA384

 © TCG 2025

objectAttributes	TPMA_OBJECT	Refer to Table 23
authPolicy	TPM2B_DIGEST	
size	UINT16	0
buffer	BYTE	Empty
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_ECDSA
scheme->details	TPMI_ALG_HASH	TPM_ALG_SHA384
curveID	TPMI_ECC_CURVE	TPM_ECC_NIST_P384
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
unique	TPMS_ECC_POINT	
x->size	UINT16	48
x->buffer	BYTE	x-coordinate
y->size	UINT16	48
y->buffer	BYTE	y-coordinate

Table 24: SPDM-identity key public structure (TPMT_PUBLIC) for ECC NIST P-384

6.1.2.2 ECC NIST P-521

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA512
objectAttributes	TPMA_OBJECT	Refer to Table 23
authPolicy	TPM2B_DIGEST	
size	UINT16	0
buffer	BYTE	Empty
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_ECDSA
scheme->details	TPMI_ALG_HASH	TPM_ALG_SHA512
curveID	TPMI_ECC_CURVE	TPM_ECC_NIST_P521
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
unique	TPMS_ECC_POINT	
x->size	UINT16	66
x->buffer	BYTE	x-coordinate
y->size	UINT16	66
y->buffer	BYTE	y-coordinate

Table 25: SPDM-identity key public structure (TPMT_PUBLIC) for ECC NIST P-521

6.2 TPM SPDM-Identity Key

6.2.1 Creation, Usage and Lifetime

A TPM SPDM-identity key is a static key that is provisioned by the TPM manufacturer and used only for SPDM session negotiation. I.e., the key is neither created nor used as an object by the TPM application layer. I.e., it is not addressable with a handle, it cannot be loaded, exported or created from the TPM API. The TPM SPDM-identity public key is readable (as opaque data) via TPM2_GetCapability, which allows it to be attested (via an audit session and TPM2_GetSessionAuditDigest).

The requirements for a TPM SPDM-identity key with respect to its creation, usage and lifetime are described below.

Creation:

 © TCG 2025

- A TPM SPDM-identity key SHALL be provisioned by the TPM manufacturer (i.e., injected or generated by the TPM during TPM manufacturing or before the key's first usage) and stored in TPM non-volatile memory.
    - If a TPM's SPDM secure session layer supports multiple (SPDM) algorithms for signature generation, then one key is required for each supported signature generation algorithm (e.g., ECDSA NIST P-256, NIST P-384 and NIST P-521).
- The TPM SPDM-identity key SHALL be generated using a FIPS-approved key pair generation method.

Usage:

- The TPM SPDM-identity key SHALL only be used by a TPM's SPDM secure session layer for the handshake in the TPM SPDM secure transport protocol, and SHALL NOT be used (by a TPM application layer) for cryptographic operations with TPM commands.

Lifetime:

- A TPM SPDM-identity key SHALL change in only the following circumstances:
    - A fresh TPM SPDM-identity key SHALL be generated by a TPM implementation with an SPDM secure session layer either upon TPM2_ChangeEPS or a vendor-defined command (required for FIPS compliance).
    - If a TPM SPDM-identity key is firmware-limited, a fresh TPM SPDM-identity key SHALL be generated by a TPM implementation with an SPDM secure session layer in response to a TPM firmware update that changes the TPM firmware version.
    - If a TPM SPDM-identity key is SVN-limited, a fresh TPM SPDM-identity key SHALL be generated by a TPM implementation with an SPDM secure session layer in response to a TPM firmware update that changes the TPM SVN.

Start of informative comment

A service is required to zeroize the TPM's SPDM-identity key in a FIPS compliant TPM. An implementation could use a vendor-specific service or might adapt the existing TPM2_ChangeEPS command. Note, however, that TPM2_ChangeEPS is likely to be available only during sanitization or de-commissioning of a platform, and is unlikely to be called during the operational lifetime of a TPM because it invalidates the TPM manufacturer's Endorsement Key (EK) certificate.

Whether a TPM SPDM-identity key is firmware-limited or SVN-limited is identified via the object attributes returned via TPM2_GetCapability (see section 6.1.1).

End of informative comment

 © TCG 2025

7 SPDM USE CASES

7.1 Use case 1: Remotely-attested channel (Server, Datacenter)

7.1.1 General

Start of informative comment

In this use case, the SPDM secure transport protocol is used to establish a remotely-attested channel (see section 4.3) between a host and its TPM.

This use case is applicable to: Server, Datacenter, and remotely administered PC Client platforms.

The following conditions apply (for detailed requirements, see Table 26):

- A 3rd party/ external verifier verifies the TPM and host identities
- There is no permanent binding between a TPM and host (only applicable to Server and Datacenter)
- TPM objects are protected through a TPM policy, which is bound to the presence of an SPDM session and to a specific SPDM Requester key (see section 5.7.2.2)

Note: For PC Client platforms, this use case is applicable e.g., to corporate client devices, where policies (applied by an IT department) are enforced remotely, or to personal client devices, where the device's state is validated before it is allowed access in an internal network (aka "bring your own device"). The PC Client manufacturer must have provided SPDM functionality in the CPU and the TPM, and the necessary supporting infrastructure must be available.

End of informative comment

7.1.2 Detailed Requirements

Start of informative comment

Table 26 summarizes the detailed requirements of this use case.

Requirements	Yes/No	Details
Conditions		
A 3rd party/ external verifier is involved	Yes	A 3rd party/ external verifier verifies the TPM and host identities that were used to establish the secure channel
TPM is swappable	Yes/ No	Yes for Server, Datacenter. No for PC Client
Binding between TPM and host is required	No	
Multiple components/ processes may communicate with the TPM concurrently	Yes	
System may have SPDM and non-SPDM-capable components/ processes talking with the TPM	Yes	E.g., platform firmware is SPDM-capable, but OS is not
Authentication		
TPM entity is protected through *authPolicy*	Yes	Objects are protected through *authPolicy* (including TPM2_PolicyTransportSPDM), such that objects cannot be authorized/ accessed outside of an SPDM session that was established with mutual signature verification using a particular Requester signing key.
Mutual signature verification	Yes	
TPM verifies the host identity	No	
TPM behavior		
TPM rejects unencapsulated TPM commands	No	TPM accepts unencapsulated TPM commands at any time

 © TCG 2025

Configuration of TPM necessary	No	
Compatibility		
Backwards compatible with non-SPDM-capable SW	Yes	Backwards combability is guaranteed since the TPM accepts unencapsulated TPM commands at any time

Table 26: Use case 1 – Detailed Requirements

End of informative comment

 © TCG 2025

8 TPM VENDOR-SPECIFIC SUPPORT

This section specifies SPDM functionality that MAY vary between TPM vendor implementations. For minimum requirements or allowed choices, see column “Description”. The column “Reporting mechanism” indicates how the supported functionality is reported by a TPM implementation with an SPDM secure session layer.

Supported functionality	Description	Reporting mechanism
SPDM version		
Supported SPDM specification version	Version 1.3 or later	*VersionNumberEntry* field of VERSION response
Resources		
Total number of supported SPDM connections	Minimum one	TPM2_GetCapability (see Note 1)
Total number of supported SPDM sessions	Minimum one	TPM2_GetCapability (see Note 1)
Algorithms & commands		
Supported algorithms	At least algorithm set • B (192 bits of security) or • C (256 bits of security) as defined in section 5.5.1.3.1 of this specification	*SupportedAlgorithms* field of CAPABILITIES response (see Note 2)
Supported SPDM capabilities	At least the Mandatory SPDM capabilities defined in section 5.5.1.2 of this specification	*Flags* field of CAPABILITIES response
Supported SPDM messages	At least the Mandatory SPDM requests and responses defined in section 5.5.1 and 5.5.2 of this specification	*Flags* field of CAPABILITIES response
Keys		
Type of TPM SPDM-Identity key	One of the following: • Neither firmware-limited nor SVN-limited • firmware-limited • SVN-limited	Object attributes returned by TPM2_GetCapability(capability = TPM_CAP_PUB_KEYS)
Service to zeroize TPM SPDM-identity key	Required for FIPS compliance (see section 5.6.4) TPM2_ChangeEPS or a vendor-defined method	Not reported (check vendor documentation)
Field Upgrade		
Support of Field Upgrade commands within SPDM session		Not reported (check vendor documentation)
Support of SPDM messages during Field Upgrade		Not reported (check vendor documentation)

Table 27: Vendor-specific support

Start of informative comment

Note 1: A TPM implementation with an SPDM secure session layer may support a TPM property to report the total number of supported SPDM connections and sessions.

 © TCG 2025

Note 2: Reporting the supported algorithms in the CAPABILITIES response requires support for SPDM “Extended capabilities” and “Large SPDM message transfer mechanism”, which is optional in the SPDM version 1.3 specification [1] and, therefore, not required by this (TCG) specification.

End of informative comment

 © TCG 2025