Document Identifier: DSP0277

Date: 2024-06-10

Version: 1.2.0

Secured Messages using SPDM Specification

Supersedes: 1.1.0

Document Class: Normative

Document Status: Published

Document Language: en-US

9 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

10 Implementation of certain elements of this standard or proposed standard may be subject to third-party patent rights, including provisional patent rights (herein “patent rights”). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third-party patent right owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners, or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third-party patent rights, or for such party’s reliance on the standard or incorporation thereof in its product, protocols, or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

11 For information about patents held by third parties which have notified DMTF that, in their opinion, such patents may relate to or impact implementations of DMTF standards, visit https://www.dmtf.org/about/policies/disclosures.

12 This document’s normative language is English. Translation into other languages is permitted.

13 CONTENTS

14 1 Foreword

15 The Platform Management Components Intercommunications (PMCI) Working Group prepared the *Secured Messages using SPDM Specification* (DSP0277).

16 DMTF is a not-for-profit association of industry members that promotes enterprise and systems management and interoperability. For information about DMTF, see https://www.dmtf.org.

17 1.1 Acknowledgments

18 DMTF acknowledges the following individuals for their contributions to this document:

- Steven Bellock — NVIDIA Corporation
- Patrick Caporale — Lenovo
- Nigel Edwards — Hewlett Packard Enterprise
- Daniil Egranov — Arm Limited
- Philip Hawkes — Qualcomm Inc.
- Brett Henning — Broadcom Inc.
- Jeff Hilland — Hewlett Packard Enterprise
- Yuval Itkin — NVIDIA Corporation
- Theo Koulouris — Hewlett Packard Enterprise
- Eliel Louzoun — Intel Corporation
- Donald Matthews — Advanced Micro Devices, Inc.
- Edward Newman — Hewlett Packard Enterprise
- Jim Panian — Qualcomm Inc.
- Scott Phuong — Cisco Systems, Inc.
- Viswanath Ponnuru — Dell Technologies
- Xiaoyu Ruan — Intel Corporation
- Nitin Sarangdhar — DMTF
- Bob Stevens — Dell Technologies
- Jiewen Yao — Intel Corporation

19 2 Introduction

20 The *Secured Messages using SPDM* specification defines the methodology that various PMCI transports can use to communicate various application data securely by utilizing SPDM. Specifically, this specification defines the transport requirements for SPDM records, which form the basis of encryption and message authentication.

21 Furthermore, this specification contains guidance and certain decisions that it defers to the binding specification, which binds Secured Messages to a specific transport. Thus, the binding specification is expected to finalize those decisions or guidance by way of normalization or recommendation. The present specification was written with PMCI transports in mind, but nothing precludes specifying bindings to other transports.

22 2.1 Document conventions

- Document titles appear in *italics*.
- The first occurrence of each important term appears in *italics* with a link to its definition.
- ABNF rules appear in a monospaced font.

23 3 Scope

24 This document defines a generic record format used to encrypt and authenticate any application data within SPDM's secure session. Also, relating to encryption, message authentication, and secure sessions, this specification further defines those areas in SPDM that the specification states are the responsibilities of the transport layer.

25 This specification requires SPDM version 1.1 or later.

26 3.1 Normative references

27 The following referenced documents are indispensable for the application of this specification. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

- DMTF DSP0274, *Security Protocol and Data Model (SPDM) Base Specification, version 1.1 or later*, https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.2.pdf
- *ISO/IEC Directives, Part 2, Principles and rules for the structure and drafting of ISO and IEC documents - 2021 (9th edition)*
- IETF RFC5116, *An Interface and Algorithms for Authenticated Encryption*, January 2008
- IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008, https://tools.ietf.org/html/rfc5234
- IETF RFC8439, *ChaCha20 and Poly1305 for IETF Protocols*, June 2018
- IETF RFC8998, *ShangMi (SM) Cipher Suites for TLS 1.3*, March 2021
- *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3, 2020-06-03 Draft*, https://datatracker.ietf.org/doc/draft-ietf-tls-dtls13/

28 3.2 Terms and definitions

29 In this document, some terms have a specific meaning beyond the normal English meaning. This clause defines those terms.

30 The terms "shall" ("required"), "shall not," "should"("recommended"), "should not" ("not recommended"), "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 7. The terms in parentheses are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Clause 7 specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

31 The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 6.

32 The terms “normative” and “informative” in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled “(informative)” do not contain normative content. Notes and examples are always informative elements.

33 The terms that DSP0274 define also apply to this document with the following exception:

- Whereas DSP0274 defines a secure session as consisting of either encryption, message authentication, or both, this document defines a secure session as consisting of both encryption and message authentication or message authentication without encryption.

34 3.3 Symbols and abbreviated terms

35 The abbreviations or notations defined in DSP0274 apply to this document.

36 4 Secured Message

37 Starting with SPDM 1.1, SPDM describes at a very high and abstract level a construct, called a record, to encrypt and authenticate data within a session. SPDM places the responsibility of the details and definition of the record onto the transport layer. The manifestation of this record in this specification is called a Secured Message.

38 A Secured Message shall only be used within a secure session. Specifically, a Secured Message can be used in any phase of a secure session, such as the session handshake phase and the application phase.

39 To support a Secured Message, an SPDM endpoint shall support message authentication. Additionally, an SPDM endpoint shall support one or more AEAD algorithms as defined in DSP0274. Finally, an SPDM Responder shall select an AEAD algorithm according to DSP0274.

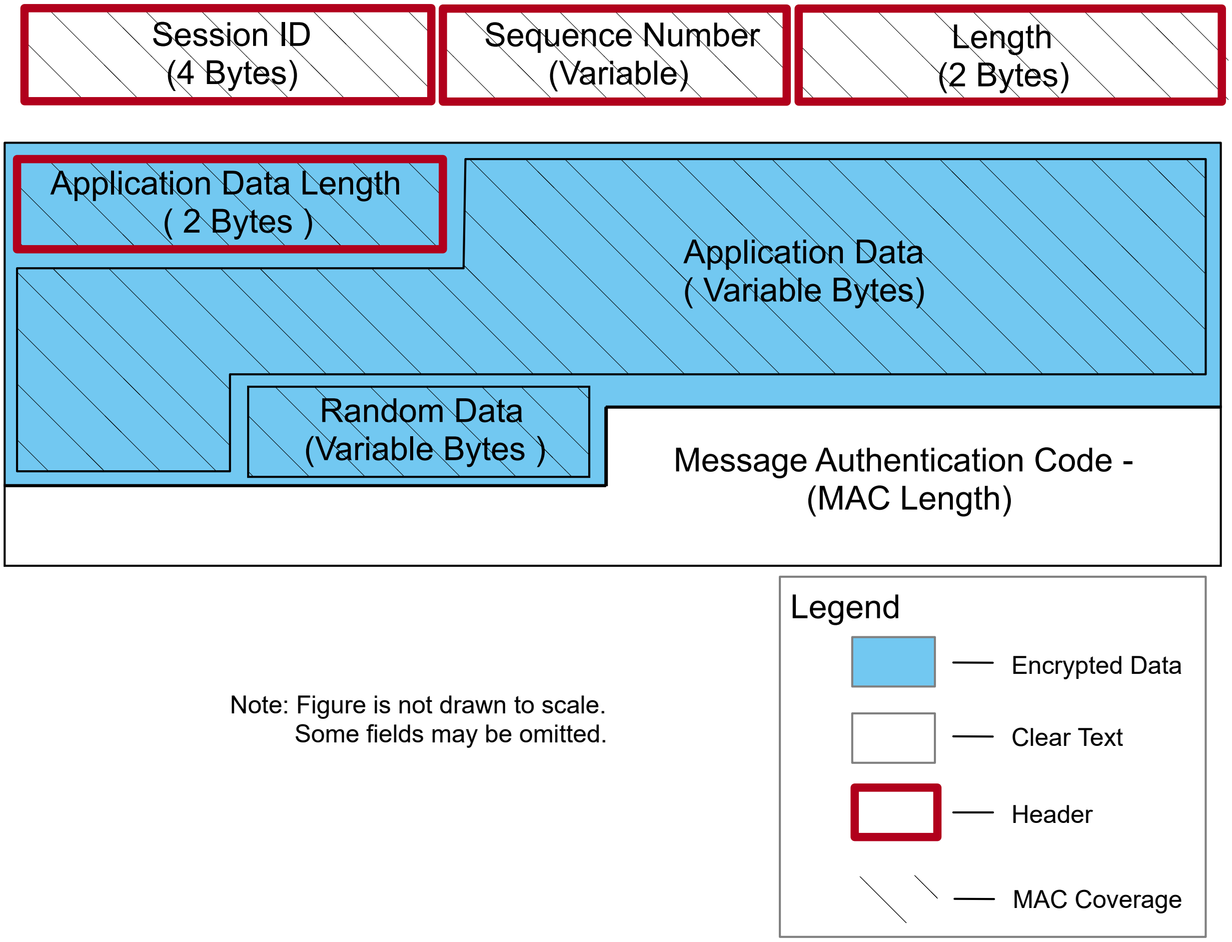
40 When Secured Messages are transmitted between entities that are not SPDM endpoints, the clauses in this specification apply unless otherwise specified by the transport binding.

41 4.1 Secured Message format

42 The Secured Message format figure illustrates the format used to encrypt or authenticate application data. Because this specification creates a single format for all application data, application data shall also include those SPDM messages that are required to be sent within a session such as `KEY_UPDATE` messages.

43 **Secured Message format**

44



45 The Secured Message fields definition table defines the exact format. This format is specific to AEAD algorithms that append the MAC to the end of the cipher text. The *Recommendation* column applies to the transport and provides a recommendation of its inclusion in the transport binding specification. A value of *Yes* in this column indicates that the respective field should be included in the transport binding specification. A value of *No* indicates the respective field should not be included unless necessary and the *Description* column will have further guidance.

46 **Secured Message fields definition**

Offset	Field	Length (bytes)	Recommendation	Description
0	Session ID	4	Yes	The Responder and Requester can use this field to bind all session information such as secrets and keys. This field shall be the same value as the `SessionID` variable in the Session-Secrets-Exchange response. To ensure forward and backward compatibility, this field shall never change and shall always be first.
4	Sequence Number	S	No	This field shall indicate all or part of the Sequence Number of the Secured Messages as described in the Per-message nonce derivation clause. See the Transmission reliability clause for further details. The transport binding specification shall specify the length `S` and further details, if any, for this field.
4 + S	Length	2	Yes	This field shall indicate the remaining length of data in Secured Messages.
6 + S	Application Data Length	2	Yes	This field shall be the length of `Application Data`.

Offset	Field	Length (bytes)	Recommendation	Description
8 + S	Application Data	P	Yes	This field shall contain either application data or in-session SPDM messages.
8 + P + S	Random Data	R	Yes	This field should contain random data of random length.
See Description	Message Authentication Code (MAC)	Variable	Yes	This field is for illustrative purposes only. The actual location and details of the MAC in the cipher text shall adhere to the AEAD specification for the selected AEAD cipher in `ALGORITHMS` response. AEAD algorithms usually append the MAC to the end of the cipher text.

47 Except for the MAC, Application Data, and Random Data fields, all fields are in little endian.

48 The purpose of the random data field is to further obfuscate the application data by hiding its real length. This prevents information from being derived from the real length of application data. In certain scenarios that do not require this obfuscation, the `Random Data` field can have a length of zero.

49 The `Length` field shall be the sum of the length of the following fields:

- Application Data Length (if present)
- Random Data (if present)
- Application Data
- MAC

50 The Field presence requirement table describes the presence requirement for each field in Secured Messages. Initially, both the Requester and Responder advertise their capabilities through `GET_CAPABILITIES` and `CAPABILITIES` messages. The two capabilities of interest are encryption (`ENCRYPT_CAP`) and message authentication (`MAC_CAP`). For a session to provide message authentication, both the Requester and Responder need to support message authentication. Lastly, for a session to provide both encryption and message authentication, both the Responder and Requester have to support both.

51 The most common denominator of capabilities between an SPDM Requester and its Responder shall determine the

capabilities for all sessions between them. SPDM does not allow a Requester and Responder to support different capabilities for each individual session. In other words, if both the Requester and Responder support both message authentication and encryption, all their sessions shall support message authentication and encryption; not one or the other, but both. Likewise, if there are no common capabilities between the two, Secured Messages shall not be supported.

52 If a session can only provide message authentication, the `Message Authentication Only` column is applicable. If a session provides both, the `Encryption and Message Authentication` column is provided. Only one column applies per session. An encryption only session shall be prohibited. If no columns apply, this specification is not applicable.

53 In the applicable column, a value of **Present** shall indicate the respective field is present in the Secured Message. A value of **Absent** shall indicate the respective field shall not be present in Secured Messages.

54 **Field presence requirement**

Field	Message Authentication Only	Encryption and Message Authentication
Length	Present	Present
Session ID	Present	Present
Sequence Number	See Transport Binding	See Transport Binding
Application Data Length	Absent	Present
Random Data	Absent	Present
Application Data	Present	Present
MAC	Present	Present

55 For some transport bindings, the Application Data field will need a specified format to ensure correct processing at the receiver. For example, if the Application Data field can carry messages from a range of protocols, the Application Data field might need a field indicating which protocol to use for processing the message in the Application Data field. If required, such formatting shall be specified by the binding specification.

56 4.2 Secured Message protection

57 Secured Messages utilize Authenticated Encryption with Associated Data (AEAD) cipher algorithms in much the same way that TLS 1.3 does. See DSP0274 for an overview of AEAD algorithms.

58 4.2.1 AEAD encryption keys and other secrets

59 SPDM's key schedule produces four major secrets that are used at certain points in the session and each secret is bound to a particular direction of transmission. The encryption keys and initialization vector (IV) are derived from these four major secrets. See Key Schedule in DSP0274 for more details.

60 4.2.2 AEAD requirements

61 This clause discusses the requirements for each parameter to the AEAD functions (`encryption_key` , `nonce` , `associated_data` , `plaintext` and `ciphertext`) depending on the capabilities of the session. See the *Application data* clause in DSP0274 for the AEAD function interfaces and more details. The references below shall be interpreted according to DSP0274 definitions.

62 In general, the MAC covers the associated data and the plain text. Specifically, the MAC covers all fields in Secured Message. The default length of the MAC shall be 16 bytes for AES-GCM, AEAD_SM4_GCM and ChaCha20-Poly1305. The transport binding can specify a different MAC length.

63 4.2.2.1 Message Authentication Only session

64 For sessions that are capable of only supporting message authentication, the `associated_data` for AEAD shall be the concatenation of the following fields in this order:

1. Session ID
2. Sequence Number (if present)
3. Length
4. Application Data

65 The fields are as defined in table Secured Message fields definition. The text to encrypt, `plaintext` , for AEAD shall be null. Consequently, the text to decrypt, `ciphertext` , shall also be null.

66 4.2.2.2 Encryption and Message Authentication session

67 The `associated_data` for AEAD shall be the concatenation of the following fields in this order:

1. Session ID
2. Sequence Number (if present)
3. Length

68 The fields are as defined in table Secured Message fields definition. The text to encrypt, `plaintext` , for AEAD shall be the concatenation of these fields in this order:

1. Application Data Length
2. Application Data
3. Random Data

69 The fields are as defined in table Secured Message fields definition. The text to decrypt, `ciphertext` , shall be the encrypted portion of the Secured Message and the MAC.

70 4.2.3 Per-message nonce derivation

71 The nonce shall never be transmitted in Secured Messages. This means that both the Responder and Requester

must internally track the nonce. To ensure proper tracking, the Requester and Responder shall follow the nonce derivation schedule laid henceforth.

72 Before the creation of the first Secured Message in the session for a given major secret and its derived encryption and IV keys, both the Responder and Requester shall start with an eight byte sequence number with a value of zero. The sequence number shall be encoded as little endian in memory, so that the least-significant byte is located at address offset `0` and the most-significant byte is located at address offset `7`. For each record, both SPDM endpoints shall follow these steps as prescribed:

1. Retrieve the `iv_length` value according to the selected AEAD cipher suite in the `ALGORITHMS` message.
2. Zero extend the sequence number by appending bytes with a value of `0` from address offset `8` to address offset `iv_length - 1`.
    - The output of this step is called the zero-extended sequence number.
3. Perform a bitwise XOR of the zero-extended sequence number with the appropriate IV derived in the SPDM key schedule.
    - The output of this step is called the per-message nonce.
4. Increment the sequence number by a value of one for the next Secured Message.

73 Because different secrets are used for different directions of data transmission, each endpoint would have to track two sequence numbers: one for the reception and the other for the transmission.

74 Lastly, when a `KEY_UPDATE` occurs, the sequence number shall reset to 0 before sending the first Secured Message using the new session keys.

75 4.2.3.1 Other per-message nonce requirements

76 A Secured Message shall not reuse a sequence number. Furthermore, an SPDM endpoint shall not send Secured Messages out of sequence. The Per-message nonce derivation clauses describes the proper sequence. This requirement does not prevent a transport from sending messages asynchronously or interleave messages as long as the transport supports message interleaving and asynchronous transfers.

77 4.2.4 Encryption requirements

78 A single Secured Message shall contain the complete ciphertext as produced by a single invocation of `AEAD_Encrypt` using the appropriate encryption key for the given direction of transmission, the appropriate per-message nonce, and the selected AEAD Cipher Suite in `ALGORITHMS`. No two or more Secured Messages shall use the same nonce. This requirement does not prevent a transport from sending messages asynchronously or interleave messages as long as the transport supports message interleaving and asynchronous transfers.

79 5 Compatibility

80 This specification is decoupled from DSP0274 to avoid unnecessary updates here whenever SPDM changes. If a tighter coupling to DSP0274 is desired, the transport binding specification should describe it.

81 Furthermore, this specification follows these rules to minimize changes between minor versions to ensure compatibility:

- Computations and other operations between different minor versions of the Secured Messages using SPDM specification should remain the same, unless security issues of lower minor versions are fixed in higher minor versions and the fixes require change in computations or other operations.
- In a newer minor version of the Secured Message using SPDM specification, a given message can be longer, bit fields and enumerations can contain new values, and reserved fields can gain functionality. Existing numeric and bit fields retain their existing definitions.

82 6 Version support

83 To advertise the supported Secured Message version of an SPDM Requester for a particular transport, the Secured Message transport binding version format table defines the fields necessary to specify the transport binding specification version of Secured Messages.

84 **Secured Message transport binding version format**

Bit	Field	Value
[15:12]	MajorVersion	Shall be the major version of the transport binding. See DSP0274 for description of major version.
[11:8]	MinorVersion	Shall be the minor version of the transport binding. See DSP0274 for description of minor version.
[7:4]	UpdateVersionNumber	Shall be the update version of the transport binding. See DSP0274 for description of Update version.
[3:0]	Alpha	Shall be the alpha version of the transport binding. For released versions, this field shall be zero. See DSP0274 for description of alpha version.

85 The transport binding specification of Secured Messages may offer a more descriptive or definitive statement on compatibility between major, minor and update versions.

86 The Secured Message transport binding version list format table describes a format to list all supported versions of the Secure Message transport binding.

87 **Secured Message transport binding version list format**

Offset	Field	Size (in bytes)	Description
0	VersionCount	1	Shall indicate the total number (T) of Secured Message transport binding versions listed in `VersionsList`
1	VersionsList	T * 2	Shall list all versions that are supported by an SPDM Requester for the given transport. The format for this field shall be the format described by the Secured Message transport binding version format table.

88 The transport binding specification shall define which version(s) of DSP0277 (this specification) it binds to. DSP0277 recommends binding to compatible versions of DSP0277 for a given version of the transport binding specification.

89 Additionally, the transport binding specification should define the values to populate in `VersionsList`. If the transport binding specification does not define the values to populate in `VersionsList`, then the version of the transport binding specification, itself, shall be the value to populate in `VersionsList`. For example, if version 1.3.4 of a transport binding specification states that it binds to version 1.0.0 of DSP0277 and does not define the values to populate in `VersionsList`, then the value used for `VersionsList` is 1.3.4. In another example, a 2.4 version of the transport binding specifically defines a value of 4.5 as the value to use for `VersionsList`. The Requester should populate `VersionsList` with all versions it supports.

90 6.1 Version selection

91 Version discovery and selection occurs during Session-Secrets-Exchange. First, the SPDM Requester shall advertise its list of supported version through the `OpaqueData` field of a Session-Secrets-Exchange request. The Requester shall use the Secured Message opaque data format to specify its list in Supported version list data format.

92 Lastly, the Responder shall select a version among the ones that are supported by the Requester and communicate the selected version in the `OpaqueData` field in the Session-Secrets-Exchange response. The Responder shall use either the Secured Message opaque data format or SPDM general opaque data format to specify its selected version in Version selection data format. The responder should select the latest supported common version. From that point on, both the SPDM Requester and Responder shall not change this version for that session. If the Responder cannot select a version, an ERROR response shall be sent with `ErrorCode=InvalidRequest`.

93 7 Transport requirements or allowances

94 This clause and subclauses describe various requirements or flexibility allowed at the transport layer.

95 7.1 Transmission reliability

96 Secured Messages rely on the transport to perform reliable lossless delivery. The transport defines the mechanisms to ensure the transmission of data, which can include retries. Furthermore, this specification expects the transport to either deliver Secured Messages in order or allow the receiver to determine the correct order of transmission of Secured Messages. In an event that transmission or reception fails, an SPDM Requester or Responder may terminate the session or restart a new one.

97 If a transport cannot provide the aforementioned characteristics, the transport binding may add the sequence number as described in Per-message nonce derivation as a field to Secured Message and provide additional guidance, if any, to properly utilize the sequence number to authenticate or decrypt the Secured Message. See DTLS 1.3 for further guidance.

98 7.2 Certain SPDM message allowances

99 If possible, the transport binding specification should take full advantage of asynchronous and bidirectional communication to allow messages such as `KEY_UPDATE` and `HEARTBEAT` to be sent directly from a Responder without any other assistance such as a sideband alerting mechanism or SPDM's `GET_ENCAPSULATED_REQUEST` mechanism. The transport binding specification shall address this.

100 7.3 ERROR response message allowances

101 Furthermore, the `ERROR` message may be sent without an SPDM request when the error code is a decryption error (`ErrorCode=DecryptError`) to indicate that the Secured Message that was received could not be decrypted or authenticated properly. In addition, both the SPDM Requester and Responder may send an `ERROR` message with `ErrorCode=DecryptError` . This is especially useful for data sent at the application layer. In other words, in this scenario, the `ERROR` response message is behaving as a response to the inability to decrypt or authenticate the received Secured Message. In the event an SPDM endpoint receives this particular error message, the SPDM endpoint should terminate the session.

102 7.4 Key update allowances

103 Unless otherwise specified by the transport, in a secure session, the SPDM endpoints can continue to transfer application data while a key update using the `KEY_UPDATE` message is in progress.

104 On the sender end, after the sender sends out a `KEY_UPDATE` (Key Operation == UpdateKey) request to the receiver

and before the sender receives a corresponding `KEY_UPDATE_ACK` response from the receiver, the sender uses the current (old) session key for protecting outgoing application data and retries of `KEY_UPDATE` (Key Operation == UpdateKey). After the sender receives the `KEY_UPDATE_ACK` response from the receiver, it deletes the old session key and starts using the new session key for protecting outgoing `KEY_UPDATE` (Key Operation == VerifyNewKey) request and application data. The first message that is protected by the new session key is the outgoing `KEY_UPDATE` (Key Operation == VerifyNewKey) request.

105 On the receiver end, the receiver keeps the old session key after sending out the `KEY_UPDATE_ACK` (Key Operation == UpdateKey) response, because the receiver may still receive messages protected by the old session key (either application data or retries of `KEY_UPDATE` (Key Operation == UpdateKey)). Those messages were sent by the sender before `KEY_UPDATE_ACK` (Key Operation == UpdateKey) was processed by the sender. In other words, after sending out the `KEY_UPDATE_ACK` (Key Operation == UpdateKey) response, the receiver may have to try decrypting an incoming message twice, with the new session key and the old session key, respectively. If the transport guarantees the order of message delivery, then the receiver deletes the old session key as soon as it receives a message protected by the new session key. If the transport does not guarantee the order of message delivery, then the earliest the receiver can delete the old session key is when it receives a message using the new session key and the longest it can keep the old session key is when all prior messages have been decrypted or messages encrypted with the old key can be considered lost. These situations should be addressed by the binding specification.

106 8 Secured Messages opaque data format

107 In many SPDM requests and response, an opaque data field exists to accommodate transport, standard organizations, or vendor-specific use cases. The Secured Message general opaque data table defines the general format for opaque data fields present during SPDM key exchange. If the selected SPDM protocol version is 1.1, then `OpaqueData` fields in the `KEY_EXCHANGE` , `KEY_EXCHANGE_RSP` , `PSK_EXCHANGE` , and `PSK_EXCHANGE_RSP` messages shall utilize the format defined by Secured Message general opaque data. This format allows an SPDM message to contain multiple vendor or standard bodies' opaque data without collision.

108 **Secured Message general opaque data table**

Offset	Field	Length (bytes)	Description
0	SpecID	4	Shall be 0x444D5446. This value is the hexadecimal representation of the string DMTF. The purpose of this field is to help distinguish opaque data defined by this specification from other opaque data.
4	OpaqueVersion	1	This field shall identify the format of the remaining bytes. This value shall be 1.
5	TotalElements	1	Shall be the total number of elements in `OpaqueList` .
6	Reserved	2	Reserved
8+	OpaqueList	Variable	Shall be a list of Opaque Elements.

109 The Opaque element table defines the format for each element in `OpaqueList` .

110 **Opaque element table**

Offset	Field	Length (bytes)	Description
0	ID	1	Shall be one of the values in the `ID` column of "Registry or standards body ID" table as defined in DSP0274.

Offset	Field	Length (bytes)	Description
1	VendorLen	1	Length in bytes of the `VendorID` field. If the data in `OpaqueElementData` belongs to a standards body, this field shall be 0. Otherwise, the data in `OpaqueElementData` belongs to the vendor and therefore, this field shall be the length indicated in the `Vendor ID` column of “Registry and standards body ID” table for the respective `ID` defined in DSP0274.
2	VendorID	VendorLen	If `VendorLen` is greater than zero, this field shall be the ID of the vendor corresponding to the `ID` field. Otherwise, this field shall be absent.
2 + `VendorLen`	OpaqueElementDataLen	2	Shall be the length of `OpaqueElementData`.
4 + `VendorLen`	OpaqueElementData	Variable	Shall be the data defined by the vendor or standards body.

Offset	Field	Length (bytes)	Description
4 + `VendorLen` + `OpaqueElementDataLen`	AlignPadding	AlignPaddingSize = 0, 1, 2 or 3	If 4 + `VendorLen` + `OpaqueElementDataLen` is not a multiple of four, this field shall be present and of the correct length to ensure 4 + `VendorLen` + `OpaqueElementDataLen` + `AlignPaddingSize` is a multiple of four. The value of this field shall be all zeros and the size of this field shall be 0, 1, 2 or 3 bytes.

111
8.1 Secured Message opaque element data format

112 The Secured Message opaque element data format implements the Opaque element for use cases specific to this specification. The Secured Message opaque element table describes the implementation.

113 **Secured Message opaque element table**

Offset	Field	Length (bytes)	Description
0	ID	1	Shall be zero to indicate DMTF.
1	VendorLen	1	Shall be zero. Note: DMTF does not have a vendor registry.
2	OpaqueElementDataLen	2	Shall be the length of the remaining bytes excluding the `AlignPadding`.
4	SMDataVersion	1	Shall identify the format of the remaining bytes. The value shall be one.
5	SMDataID	1	Shall be the identifier for the Secured Message data type. Later clauses of this specification describe the allowed values.

Offset	Field	Length (bytes)	Description
6	SMData	SMDataLen	Shall be the data corresponding to `SMDataID`. Later clauses of this specification describe this format.
6 + `SMDataLen`	AlignPadding	AlignPaddingSize = 0, 1, 2 or 3	If 6 + `SMDataLen` is not a multiple of four, this field shall be present and of the correct length to ensure 6 + `SMDataLen` + `AlignPaddingSize` is a multiple of four. The value of this field shall be all zeros and the size of this field shall be 0, 1, 2 or 3 bytes.

114 8.1.1 Version selection data format

115 The Version selection data format table implements the Secured Message opaque element to communicate the selected Secured Message version. This data type shall only be allowed in a Session-Secrets-Exchange Response. An SPDM Responder populates this information.

116 **Secured Message version selection data format table**

Offset	Field	Length (bytes)	Description
0	ID	1	Shall be zero to indicate DMTF.
1	VendorLen	1	Shall be zero. Note: DMTF does not have a vendor registry.
2	OpaqueElementDataLen	2	Shall be four.
4	SMDataVersion	1	Shall identify the format of the remaining bytes. The value shall be one.
5	SMDataID	1	Shall be a value of zero to indicate Secured Message version selection.

Offset	Field	Length (bytes)	Description
6	SelectedVersion	2	Shall be the selected Secured Message Version. See Secured Message transport binding version format for the format of this field.

117 ### 8.1.2 Supported version list data format

118 The Supported version list data format table implements the Secured Message opaque element to list all the supported Secured Message transport binding versions. This data type shall only be allowed in a Session-Secrets-Exchange Request. An SPDM Requester populates this information.

119 **Supported version list data format**

Offset	Field	Length (bytes)	Description
0	ID	1	Shall be zero to indicate DMTF.
1	VendorLen	1	Shall be zero. Note: DMTF does not have a vendor registry.
2	OpaqueElementDataLen	2	Shall be the length of the remaining bytes excluding the `AlignPadding`.
4	SMDataVersion	1	Shall identify the format of the remaining bytes. The value shall be one.
5	SMDataID	1	Shall be a value of one to indicate Supported version list.
6	SecuredMsgVers	SMDataLen = (T * 2) + 1	Shall be the format described in Secured Message transport binding version list format.
6 + `SMDataLen`	AlignPadding	AlignPaddingSize	See `AlignPadding` field in Secured Message opaque element table.

120 9 SPDM general opaque data format

121 In SPDM version 1.2 and later, SPDM defines a general opaque data format (`OpaqueDataFmt1`). When the SPDM requester selects SPDM version 1.2 or later for the SPDM connection, only `OpaqueDataFmt1` is supported. The Requester and Responder shall support `OpaqueDataFmt1` and the Responder shall select `OpaqueDataFmt1` in the `ALGORITHMS` message.

122 The SPDM Opaque element format is compatible with Secured Message opaque element format. The format for version selection and advertising the list of supported versions can be reused. The Version selection data format table shall be used to communicate the selected Secured Message version by the SPDM responder and the Supported version list data format shall be used to communicate the list of supported Secured Message versions by the Requester. In these two formats, the `SMDataVersion` , `SMDataID` , `SelectedVersion` and `SecuredMsgVers` fields map to the `OpaqueElementData` field of the Opaque Element table that SPDM defines.

123 The Secured Message general opaque data table shall not be used because SPDM specification version 1.2 and later defines its own.

124 10 ANNEX A (informative) Sequence number layout

125 In this example the 8-byte sequence number with value `0x01FF02EE03DD04CC`, located at address offsets `0x0` through `0x7`, is zero-extended for the Per-message nonce derivation. `iv_length` is `12` and address offsets `0x8` through `0xB` contain the extended zeroes.

Address Offset	Value
0x0	0xCC
0x1	0x04
0x2	0xDD
0x3	0x03
0x4	0xEE
0x5	0x02
0x6	0xFF
0x7	0x01
0x8	0x00
0x9	0x00
0xA	0x00
0xB	0x00

126 11 ANNEX B (informative) change log

127 11.1 Version 1.0.0 (2020-09-18)

- Initial release

128 11.2 Version 1.1.0 (2022-03-28)

- Minor typographical fixes
- Clarified `DecryptError` usage when a session implements message authentication only in ERROR response message allowances.
- Clarified which versions get populated in `VersionsList` in Secured Message transport binding version list format.
- New:
  - Added SPDM general opaque data format clauses.
  - Allow for two general opaque data format in Version selection.
  - Add Key update allowances clauses.

129 11.3 Version 1.2.0 (2024-06-10)

- Minor typographical fixes
- Clarified that the transport must bind to one version of this specification or a set of compatible versions of this specification.
- Clarified that a responder should select the highest supported version of opaque data during version negotiation.
- Clarification on the fields in AEAD construction.
- Clarified that the Random Data field is not little endian.
- Specified the AEAD sequence number as little endian.

130 12 Bibliography

131 DMTF DSP4014, *DMTF Process for Working Bodies 2.6*, https://www.dmtf.org/sites/default/files/standards/documents/DSP4014_2.6.pdf