TRUSTED® COMPUTING GROUP

SPECIFICATION

TCG Storage Feature Set: Epoch Key Purge (EKP)

Version_1

August 7, 2026

Contact: admin@trustedcomputinggroup.org

PUBLISHED

© TCG 2026

 © TCG 2026

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.

ACKNOWLEDGEMENT

The TCG wishes to thank all those who contributed to this specification. This document builds on work done in various working groups in the TCG and industry at large.

NAME	COMPANY
Giri Mandyam	AMD
Jaleel Kazi	Dell, Inc.
Charles Kunzman	Google Inc.
Jeff Anderson	Google Inc.
Joy Shukla	Google Inc.
Srini Narayanamurthy	Google Inc.
Zach Halvorsen	Google Inc.
Luis Freeman	IBM
James Borden	Kioxia Corporation
John Geldman	Kioxia Corporation
Taku Kato	Kioxia Corporation
Frederick Knight	Kioxia Corporation
Paul Suhler	Kioxia Corporation
Kensaku Yamaguchi	Kioxia Corporation
Alan Arnold	Lenovo Inc.
James Leighton	Lenovo Inc.
Roney Michaely	Lenovo Inc.
Jenna White	Lenovo Inc.
Alon Cohen	Microchip Technology, Inc.
Chander Kavalipati	Microchip Technology, Inc.
Himangini Agrawal	Micron Technology, Inc.
Bharath Madanayakanahalli Gururaj	Micron Technology, Inc.
Steve Schauer	Micron Technology, Inc.
Robert Strong	Micron Technology, Inc.
Sridhar Balasubramanian	Balasubramanian NetApp
Gwangbae Choi	Samsung Semiconductor Inc.
Eric Hibbard	Samsung Semiconductor Inc.
SeungJae Lee	Samsung Semiconductor Inc.
Yoni Shternhell	Sandisk
Anthony Duran	Seagate Technology
Festus Hategekimana	Solidigm
John Mathews	Solidigm
Patrick Hery	Toshiba Corporation
Jithendra Balasubramanian	Western Digital Technologies
Jithendra Bethur	Western Digital Technologies
Joseph Chen	ULINK
Jim Hatfield	Invited Expert
Michael McDonnell	Invited Expert
Chandra Nelogal	Invited Expert

 © TCG 2026

CONTENTS

ACKNOWLEDGEMENT **3**

1 SCOPE **8**

1.1 KEY WORDS 8
1.2 STATEMENT TYPE 8

2 INTRODUCTION AND CONCEPTS **9**

2.1 DOCUMENT PURPOSE 9
2.2 INTENDED AUDIENCE 9
2.3 CONVENTIONS 9
2.3.1 Font Conventions *9*
2.3.2 Security Provider (SP) Table Cell Symbols Legend *9*
2.3.3 List Conventions *11*
2.3.3.1 Lists Overview 11
2.3.3.2 Unordered Lists 11
2.3.3.3 Ordered Lists 11
2.3.4 Numbering Conventions *12*
2.3.5 Bit Conventions *12*
2.3.6 Number Range Conventions *12*
2.3.7 Specify and Indicate Conventions *12*
2.4 DOCUMENT REFERENCES 13
2.4.1 Document Precedence *13*
2.4.2 Normative References *13*
2.4.3 Informative References *14*
2.5 DEPENDENCIES ON OTHER FEATURE SETS 14
2.6 INTERACTIONS WITH OTHER FEATURE SETS 14
2.7 DEFINITION OF TERMS 14

3 EPOCH KEY PURGE (EKP) FEATURE SET OVERVIEW **17**

3.1 EKP FEATURE SET FUNCTIONALITY AND DEPENDENCIES 17
3.2 EKP FEATURE SET USE CASES AND THREATS 18
3.3 MANAGING EPOCH KEYS 18
3.4 ATTESTABLE EKP OPERATIONS 24

4 SECURITY SUBSYSTEM CLASS (SSC) SPECIFIC FUNCTIONALITY **26**

4.1 GENERAL 26
4.2 METHODS 26
4.2.1 New Methods *27*
4.2.1.1 EKP_Program_OTP_HEK 27
4.2.1.1.1 Parameter Descriptions 28
4.2.1.1.2 Returned Value Descriptions 28
4.2.1.1.3 EKP_Program_OTP_HEK Method Operation 28
4.2.1.2 EKP_Sanitize_OTP_HEK 31
4.2.1.2.1 Parameter Descriptions 32
4.2.1.2.2 Returned Value Descriptions 32
4.2.1.2.3 EKP_Sanitize_OTP_HEK Method Operation 32
4.2.1.3 EKP_Enable_Permanent_HEK 33

 © TCG 2026

4.2.1.3.1 Parameter Descriptions 34
4.2.1.3.2 Returned Value Descriptions 34
4.2.1.3.3 EKP_Enable_Permanent_HEK Method Operation 34
4.2.1.4 EKP_Program_SEK 36
4.2.1.4.1 Parameter Descriptions 36
4.2.1.4.2 Returned Value Descriptions 36
4.2.1.4.3 EKP_Program_SEK Method Operation 36
4.2.1.5 EKP_Sanitize_SEK 37
4.2.1.5.1 Parameter Descriptions 38
4.2.1.5.2 Returned Value Descriptions 38
4.2.1.5.3 EKP_Sanitize_SEK Method Operation 38
4.2.1.6 EKP_Attested_Report 39
4.2.1.6.1 Parameter Descriptions 40
4.2.1.6.2 Returned Value Descriptions 40
4.2.1.6.3 EKP_Attested_Report Method Operation 40
4.2.2 Modified Methods 41
4.2.2.1 Set 41
4.2.2.1.1 Set Method Operations 42
4.2.2.2 Activate 45
4.2.2.3 User Data Removal Methods 45
4.3 TABLES 45
4.3.1 New Tables 45
4.3.1.1 EpochKeyPurge (Object Table) 45
4.3.1.1.1 UID 46
4.3.1.1.2 Name 46
4.3.1.1.3 CommonName 46
4.3.1.1.4 EKP_Termination_Policy 46
4.3.1.1.5 EKP_Allowed 46
4.3.1.1.6 MaxSupported_HEK_Sanitizations 47
4.3.1.1.7 Remaining_HEK_Sanitizations 47
4.3.1.1.8 HEK_State 48
4.3.1.1.9 HEK_Size 49
4.3.1.1.10 SEK_State 49
4.3.1.1.11 SEK_Size 50
4.3.1.1.12 HEK_State_List 50
4.3.1.2 EKPCertList (Object Table) 51
4.3.1.2.1 UID 51
4.3.1.2.2 Name 51
4.3.1.2.3 CommonName 51
4.3.1.2.4 Max_Certificates 51
4.3.1.2.5 EKP_Cert_List 51
4.3.2 Modified Tables 51
4.4 TYPES 52
4.4.1 New Types 52
4.4.1.1 Restricted Type 52
4.4.1.1.1 ekp_certificates_object_uidref 52
4.4.2 Modified Types 52
5 FEATURE SET REQUIREMENTS 53
5.1 REQUIREMENTS OVERVIEW 53
5.2 LEVEL 0 DISCOVERY 53

5.2.1 Epoch Key Purge Feature Descriptor (Feature Code = 0x040D) 53
5.2.1.1 Feature Code 53
5.2.1.2 Feature Descriptor Version Number 53
5.2.1.3 Feature Set Minor Version Number 53
5.2.1.4 Length 54
5.2.1.5 Maximum Number of HEK Sanitizations Supported 54
5.2.1.6 SD Reset Policy for OTP HEK 54
5.3 ADMIN SP 54
5.3.1 Tables Preconfiguration 54
5.3.1.1 MethodID 54
5.3.1.2 EpochKeyPurge table 56
5.3.1.3 EKPCertList table 57
5.3.1.4 Table Table 58
5.3.1.5 Access Control Element (ACE) 59
5.3.1.6 AccessControl 60
5.3.1.7 Certificates 65
5.3.1.8 CertificateData 66
5.4 ADDITIONAL SPS 66
5.5 EKP REPORT USING CBOR ENCODING 67
5.5.1 CBOR Encoding of EKP Evidence 67
5.5.2 COSE Protection of EKP Evidence 70

 © TCG 2026

List of Tables

Table 1 – Security Provider Table Legend 10
Table 2 OTP_Program_Policy Table Description 28
Table 3 Report Table Description 40
Table 4 EpochKeyPurge Table Description 45
Table 5 EKP_Termination_Policy Column Description 46
Table 6 EKP_Allowed Column Description 47
Table 7 HEK_State Table Description 48
Table 8 SEK_State Table Description 49
Table 9 HEK_State_List Table Description 50
Table 10 EKPCertList Table Description 51
Table 11: ekp_certificates_object_uidref 52
Table 12: Level 0 Discovery – Epoch Key Purge Feature Descriptor 53
Table 13: Feature Set Minor Versions 53
Table 14: SD Reset Policy for OTP HEK Reset Types 54
Table 15: AdminSP - MethodID Table Preconfiguration 56
Table 16: AdminSP - EpochKeyPurge Table Preconfiguration 56
Table 17: AdminSP - EKPCertList Table Preconfiguration 57
Table 18: AdminSP - Table Table Preconfiguration 58
Table 19: AdminSP - ACE Table Preconfiguration 59
Table 20: AdminSP - AccessControl Table Preconfiguration 61
Table 21: AdminSP - Certificates Table Preconfiguration 66

List of Figures

Figure 1 EKP Mode Transitions 19
Figure 2 Typical Epoch Key State Transitions 20
Figure 3 Typical EKP Mode Operations 22
Figure 4 Permanent HEK 22
Figure 5 Corrupted HEK Scenarios 23
Figure 6 HEK Operations Dependent on Resets 24

 © TCG 2026

1 Scope

This specification defines the Epoch Key Purge (EKP) Feature Set.

1.1 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.

1.2 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 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 2026

2 Introduction and Concepts

2.1 Document Purpose

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

2.2 Intended Audience

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

2.3 Conventions

2.3.1 Font Conventions

Names of methods and Security Provider (SP) tables are in `Courier New font` (e.g., the `Set` method, the `Locking` table). This convention does not apply to method and table names appearing in headings or captions.

Hexadecimal numbers are in `Courier New font`.
All other text is in the Arial font.

2.3.2 Security Provider (SP) Table Cell Symbols Legend

The legend in Table 1 defines the SP table cell symbols used to communicate Read-Write and Access Control attributes of a cell. These symbols are informative only. Unless otherwise indicated, the table cell content is normative. If a table cell is empty or blank (regardless of symbols), then the contents of that cell are not specified in this specification. The contents might be specified in another specification.

Start of Informative Comment

As specified in the TCG Storage Architecture Core Specification (see [2]) section 3.4.2 and section 5.3.4.3, Access Control limits the methods that can be processed on an SP, a table, or on specific rows and columns of a table. Access Control is specified in layers. The top layer of the mechanism is an Access Control List (ACL). ACLs contain lists of Access Control Elements (ACEs). ACEs contain Boolean combinations of authorities. ACLs and ACEs bind methods to the authorities that are permitted to invoke them. When the state of authorities specified in an ACE Boolean expression are satisfied as described in the TCG Storage Architecture Core Specification (see [2]) section 3.4.2.1 (for example, the ACE Boolean expression resolves to True on successful authentication), the ACE permits methods to be invoked on an SP, a table, or on specific rows and columns of a table based on the ACL that contains the ACE.

End of Informative Comment

The “R-W” column in Table 1 describes if the contents of an SP table cell are read-only, writeable, or not required (NR). This cell property can be determined through the mapping of methods that provide read or write access to the cell, and an ACL in the AccessControl table (see TCG Storage Architecture Core Specification [2] section 5.3.4.3). If a cell is readable, then there exists a method and associated ACE definition to retrieve the contents of the cell. If a cell is writeable, then there exists a method and associated ACE definition to modify the contents of the cell. There MAY be other conditions that prevent a method from being invoked on a table cell outside the scope of the Access Control specified in the AccessControl table.

The “Value” column in Table 1 describes if the contents of an SP table cell are specified by the feature set, vendor unique (VU), or not required (NR). When a cell’s content is VU, the cell contents contain the text “VU” and access control for the cell is defined by the feature set.

 © TCG 2026

The “Access Control” column in Table 1 describes if the access control associated with an SP table cell is user personalizable, fixed, or not defined (ND). Access Control for a cell is personalizable if the Boolean expression of the ACEs associated with the cell are writable. Access Control for a cell is fixed if the Boolean expression of the ACEs associated with the cell are not writeable. Access Control for a cell is not defined (ND) if there does not exist an ACE that is associated with the cell. If the cell contents are (NR), the `Get` method MAY omit the column from a method response. The behavior of `AddACE` and `RemoveACE` methods with respect to ACLs is not defined.

Table 1 – Security Provider Table Legend

Table Cell Symbols Legend	R-W	Value	Access Control	Comment
Read-only Fixed Access Control (RF)	Read-only	Epoch Key Purge (EKP) Feature Set specified	Fixed	• Cell content is Read-Only. • Access control is fixed. • Value is specified by the Epoch Key Purge (EKP) Feature Set
Read-only Fixed Access Control-VU (RFV)	Read-only	VU	Fixed	• Cell content is Read-Only. • Access Control is fixed. • Values are Vendor Unique (VU). A minimum or maximum value might be specified.
Undefined (UD)	Not Defined (ND)	Not Required (NR)	Not Defined (ND)	• Cell content is (NR). • Access control is not defined (ND). • Any text in table cell is informative only. • A `Get` MAY omit this column from the method response.
Writable Personalizable Access Control (WP)	Write	Preconfigured/ user personalizable, or VU	Preconfigured, user personalizable	• Cell content is writable. • Access control is personalizable • `Get` Access Control is not described by this symbol • Cell contents preconfiguration for VU column is “VU”.

 © TCG 2026

Table Cell Symbols Legend	R-W	Value	Access Control	Comment
Writable Fixed Access Control (WF)	Write	Preconfigured/ user personalizable, or VU	Fixed	• Cell content is writable. • Access control is fixed. • `Get` Access Control is not described by this symbol • Cell contents preconfiguration for VU column is “VU”.

2.3.3 List Conventions

2.3.3.1 Lists Overview

Lists are associated with an introductory paragraph or phrase and are numbered relative to that paragraph or phrase (i.e., all lists begin with an a) or 1) entry).

Each item in a list is preceded by an identification with the style of the identification being determined by whether the list is intended to be an ordered list or an unordered list.

If the item in a list is not a complete sentence, the first word in the item is not capitalized. If the item in a list is a complete sentence, the first word in the item is capitalized.

Each item in a list ends with a semicolon, except the last item, which ends in a period. The next to the last entry in the list ends with a semicolon followed by an “and” or an “or” (i.e., “…; and”, or “…; or”). The “and” is used if all the items in the list are required. The “or” is used if only one or more items in the list are required.

2.3.3.2 Unordered Lists

An unordered list is one in which the order of the listed items is unimportant (i.e., it does not matter where in the list an item occurs as all items have equal importance). In an unordered list, each item starts with a lowercase letter followed by a close parenthesis. If it is necessary to subdivide a list item further with an additional unordered list (i.e., have a nested unordered list), then the nested unordered list is indented and each item in the nested unordered list starts with an uppercase letter followed by a close parenthesis.

The following is an example of an unordered list with a nested unordered list:

EXAMPLE - The following are the items for the assembly:
a) a box containing:
    A) a bolt;
    B) a nut; and
    C) a washer;
b) a screwdriver; and
c) a wrench.

2.3.3.3 Ordered Lists

An ordered list is one in which the order of the listed items is important (i.e., item n is required before item n+1). In an ordered list, each item starts with a Western-Arabic numeral followed by a close parenthesis. If it is necessary to

subdivide a list item further with an additional unordered list (i.e., have a nested unordered list), then the nested unordered list is indented and each item in the nested unordered list starts with an uppercase letter followed by a close parenthesis.

The following is an example of an ordered list with a nested unordered list:

EXAMPLE - The following are the instructions for the assembly:
1) remove the contents from the box;
2) assemble the item;
   A) use a screwdriver to tighten the screws; and
   B) use a wrench to tighten the bolts;
   and
3) take a break.

2.3.4 Numbering Conventions

A binary number is represented in this specification by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lowercase b (e.g., 0101b). Underscores or spaces might be included between characters in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this specification by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the uppercase English letters A through F immediately preceded by "0x". Underscores or spaces might be included between characters in hexadecimal number representations to increase readability or delineate field boundaries (e.g., 0xFD8C FA23 or 0x0B_FD8C_FA23). Hexadecimal numbers are in Courier New font.

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 not immediately followed by a lowercase b or lowercase h (e.g., 25). This standard uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space; and
c) the thousands separator is used in both the integer portion and the fraction portion of a number.

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., 666.$\bar{6}$ **Error! Bookmark not defined.**means 666.666 666… or 666 2/3, and 12.$\overline{142857}$ means 12.142 857 142 857… or 12 1/7).

2.3.5 Bit Conventions

Name (n:m), where n is greater than m, denotes a set of bits (e.g., Feature (7:0)).

2.3.6 Number Range Conventions

p..q, where p is less than q, represents a range of numbers (e.g., words 100..103 represents words 100, 101, 102, and 103).

2.3.7 Specify and Indicate Conventions

In a given message, command, or other information exchange between a requestor (e.g., a host) and a responder (e.g., an SD):

a) the requestor specifies information in the request; and
b) the responder indicates information in the response.

2.4 Document References

2.4.1 Document Precedence

If there is a conflict between this specification and any other reference, then the precedence is (where a lower number indicates higher precedence):

1. This specification;
2. Normative references (see section 2.4.2).

Each normative reference and informative reference might specify its own document precedence.

2.4.2 Normative References

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

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

[3] Trusted Computing Group (TCG), “TCG Storage Interface Interactions Specification”, Version 1.20 or later

[4] Trusted Computing Group (TCG), “TCG Storage Security Subsystem Class: Opal”, Version 2.30 or later

[5] Trusted Computing Group (TCG), “TCG Storage Security Subsystem Class (SSC): Key Per I/O”, Version 1.0 or later

[6] Trusted Computing Group (TCG), “TCG Storage Security Subsystem Class: Enterprise”, Version 1.01 or later

[7] Trusted Computing Group (TCG), “TCG Storage Security Subsystem Class: Opalite Specification”, Version 1.00 or later

[8] Trusted Computing Group (TCG), “TCG Storage Security Subsystem Class: Ruby Specification”, Version 1.00 or later

[9] NIST, Special Publication 800-88, 2025, “Guidelines for Media Sanitization”. Rev. 2, Available: https://doi.org/10.6028/NIST.SP.800-88r2

[10] Internet Engineering Task Force, "Concise Binary Object Representation (CBOR)," [Online]. Available: https://www.rfc-editor.org/rfc/rfc8949.html.

[11] Internet Engineering Task Force, “CBOR Object Signing and Encryption (COSE): Structures and Process,” [Online]. Available: https://www.rfc-editor.org/rfc/rfc9052.html

[12] Internet Engineering Task Force, “CBOR Object Signing and Encryption (COSE): Initial Algorithms,” [Online]. Available: https://www.rfc-editor.org/rfc/rfc9053.html

[13] Internet Engineering Task Force, IETF. (2023-02-23). RFC 9334. *Remote Attestation Procedures Architecture*. Available online: https://datatracker.ietf.org/doc/html/rfc9334/

[14] Internet Assigned Numbers Authority, IANA. (2021-12-15). *Concise Binary Object Representation (CBOR) Tags*. Available online: https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml

[15] Internet Engineering Task Force, IETF. (2019-June). RFC 8610. *Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data*

[16] Internet Engineering Task Force, IETF. *CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates,* [Online]. Available: https://www.rfc-editor.org/rfc/rfc9360.html

[17] IANA, "Concise Binary Object Representation (CBOR) Tags", Available: https://www.iana.org/assignments/cbor-tags

[18] ISO/IEC 19790, “Information security, cybersecurity and privacy protection — Security requirements for cryptographic modules”

[19] IETF RFC 5280, 2008, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”

[20] Trusted Computing Group (TCG), “Implicit Identity Based Device Attestation”, Version 1.0

2.4.3 Informative References

This feature set does not include any informative references.

2.5 Dependencies on Other Feature Sets

This feature set does not depend upon any other feature sets.

2.6 Interactions with Other Feature Sets

This feature set does not interact with any other feature sets.

2.7 Definition of Terms

Term	Definition
CBOR	Concise Binary Object Representation
CDDL	Concise Data Definition Language
cryptographically bound	The MEK is protected in such a way that without knowledge of the plaintext HEK and SEK, recovery of the plaintext MEK is infeasible.
COSE	CBOR Object Signing and Encryption
EKP	Epoch Key Purge
EKP evidence	Metadata that provides information on the state of the SD’s use of the EKP feature set
EKP metadata	Information contained in the `EpochKeyPurge` table, which includes epoch state details and other related information
EKP mode	A special state in which an SD constrains permitted operations while using the methods defined in this feature set. The `Set` method is used to transition into the EKP mode and to exit the EKP mode.
EPK	Epoch Protection Key

 © TCG 2026

Term	Definition
epoch key	Either the hard epoch key or soft epoch key specified in this document.
Epoch Protection Key	A symmetric key (e.g., a key-encrypting key or symmetric key-derivation key) derived from the cryptographic combination of the epoch keys. This epoch protection key (EPK) is used as part of the cryptographic binding of the symmetric keys, which are critical to the data at rest encryption capabilities of the SD, to the epoch keys.
Hard Epoch Key	Symmetric key-derivation key, used for user data sanitization, that is stored in a secure, one-time programmable area of the storage device. For this feature set, the hard epoch key is used in conjunction with soft epoch key.
HEK	Hard Epoch Key
key sanitization	Controlled sanitization of a key as described in ISO/IEC 19790 [18] (e.g., overwriting with all zeros, all ones, or random data).
Media Encryption Key	Symmetric data encryption key [9] used to encrypt and/or decrypt data on the storage media.
MEK	Media Encryption Key
ND	Not Defined.
NR	Not Required.
OTP	One-time Programmable
OTP bits	Non-volatile storage that can be written exactly once after manufacturing and then permanently locked. This storage can involve the use of error correction codes and/or redundancy (e.g., triple redundancy used with majority voting to determine the correct bit value).
OTP slot	A collection of OTP bits sufficient to store specific data. For this feature set, this data is the contents of an HEK.
permanent HEK	A static key or a well-known constant (e.g., all zeroes or all ones) that is used as the HEK once all the OTP slots have been consumed. The permanent HEK is not expected to contribute significantly to the security of the SD and the permanent HEK cannot be sanitized.
SD	Storage Device
SEK	Soft Epoch Key

Term	Definition
Soft Epoch Key	Symmetric key-derivation key, used for user data sanitization, that is stored in secure, rewritable flash. For this feature set, the soft epoch key is used in conjunction with hard epoch key.
SP	Security Provider
SSC	Security Subsystem Class. SSC specifications describe profiled sets of TCG functionality
symmetric key encryption	Encipherment that uses a symmetric key algorithm along with a single key (e.g., symmetric data encryption key or symmetric key wrapping key) that is used for both encryption and decryption operations
TPer	The TCG security subsystem within a Storage Device

 © TCG 2026

3 Epoch Key Purge (EKP) Feature Set Overview

3.1 EKP Feature Set Functionality and Dependencies

Start of Informative Comment

The goal of Epoch Key Purge (EKP) Feature Set is to provide an attestable, purge storage sanitization technique that is based on a standards-based cryptographic erase technique [9] for Storage Devices (SD). This purge sanitization technique is independent of all TCG Storage Security Subsystem Classes (SSC) but can be used in conjunction with SSC user data removal methods.

Key Features:

- Feature Discoverability: A feature descriptor in Level 0 Discovery response data to allow a host to discover whether a SD supports the EKP feature set.
- Epoch Keys: The SD's symmetric keys associated with data at rest encryption are cryptographically bound to a pair of symmetric key-derivation keys that are used jointly (cryptographically combined) to provide additional storage sanitization capabilities and assurances.
- New Object Tables:
    - `EpochKeyPurge`
    - `EKPCertList`
- Sanitization Lifecycle: New TCG storage methods to allow a host to manage the lifecycle of `EpochKeyPurge` object within the SD.
- Attested Reporting: New TCG storage method to produce an attested report for the state of the epoch keys that can be used after EKP methods to verify operations as well as to view the history of OTP HEK sanitizations over the life of the SD
- Backward Compatibility: If HEK and SEK are not explicitly configured, the EKP feature set introduces no changes to the behavior defined in TCG SSC specifications or other feature sets.
- SSC Compatibility: The EKP feature set can be used in conjunction with existing TCG SSC user data removal methods to provide additional assurances for sanitizing user data on an SD.

To implement the EKP feature set functionality, an SD needs the following:

- The SD has strong hardware-based, symmetric key encryption capabilities that are always on. This encryption capability establishes the security strength and key sizes used (e.g., an XTS-AES-256 encryption engine uses a pair of 256-bit keys and has a target security strength of 256-bits).
- The SD employs two epoch keys (symmetric key-derivation keys) that are used jointly (cryptographically combined) to derive a symmetric key (often a key-encrypting key), referred to as the epoch protection key (EPK), that is critical to the data at rest encryption capabilities of the SD.
- The SD has secure hardware-based, one-time programmable (OTP) bits (e.g., fuses) that are combined into slots that are used to store one of the epoch keys, referred to as the hard epoch key (HEK) because of its hardware relationship.
- The SD OTP bits can be sanitized, as described in ISO/IEC 19790 [18] (e.g., overwriting with all zeros, all ones, or random data).
- The SD has secure, rewritable storage (e.g., in persistent flash memory) to store a second epoch key, referred to as the soft epoch key (SEK).

- The SD uses the EPK, in part or in whole, to protect (wrap) or to derive media encryption keys (MEK) used by the SD to encrypt or decrypt user data written to the SD.
- Controlled sanitization of the HEK and/or SEK renders data unrecoverable (i.e., the SD performs cryptographic erase as a purge sanitization technique [9]).
- The SD has one or more certificates available for the purpose of signing EKP evidence that is returned to a host.

End of Informative Comment

3.2 EKP Feature Set Use Cases and Threats

Start of Informative Comment

From the perspective of this feature set, the asset to be protected is user data stored at rest on an SD. The adversary profile, seeking to gain access to this user data, includes sophisticated threat actors, including nation states, organized crime, etc.

Threats:

- Exposure of user data written to a drive that cannot be removed with a TCG user data removal method (e.g., Revert method in the TCG Opal SSC [4]).
- Compromised SD firmware that changes software-based sanitization behaviors.

Assumptions for using the EKP feature set methods:

- The SD has no active SSC (e.g., TCG Opal SSC [4] or TCG Key per I/O SSC [5]).
- The SD has no active locking (e.g., there are no non-global locking objects with non-zero RangeLength, or any locking object, including the global range locking object, with WriteLockEnabled = True or ReadLockEnabled = True).

Use cases:

- OTP-based storage sanitization of SDs prior to disposal.
- OTP-based storage sanitization of SDs for reuse.
- OTP-based storage sanitization of SD transitioning between use of different TCG SSCs (e.g., Opal and Key Per I/O).
- Attestable sanitization and verification of previous OTP-based sanitizations.

End of Informative Comment

3.3 Managing Epoch Keys

Start of Informative Comment

The EKP feature set manages the epoch keys (HEK and SEK) throughout their lifecycle. Because of their ability to permanently sanitize data, most of the operations associated with the epoch keys can only be performed when the SD has transitioned into a special mode, known as the EKP mode (see Figure 1), which is controlled by the `Set` method.

Transitioning into the EKP mode also causes the SD to restrict permitted operations to those that are permitted during user data removal methods [3]. Any attempted operation that is not permitted in the EKP mode (e.g., read and write operations) results in a failure with a status of Data Protection Error. Transitioning out of the EKP mode is only possible when both the HEK and SEK are programmed.

Figure 1 also shows that the SD can remain in the EKP mode indefinitely, which means the SD cannot be used for user data reads/writes until the SD exits the EKP mode.

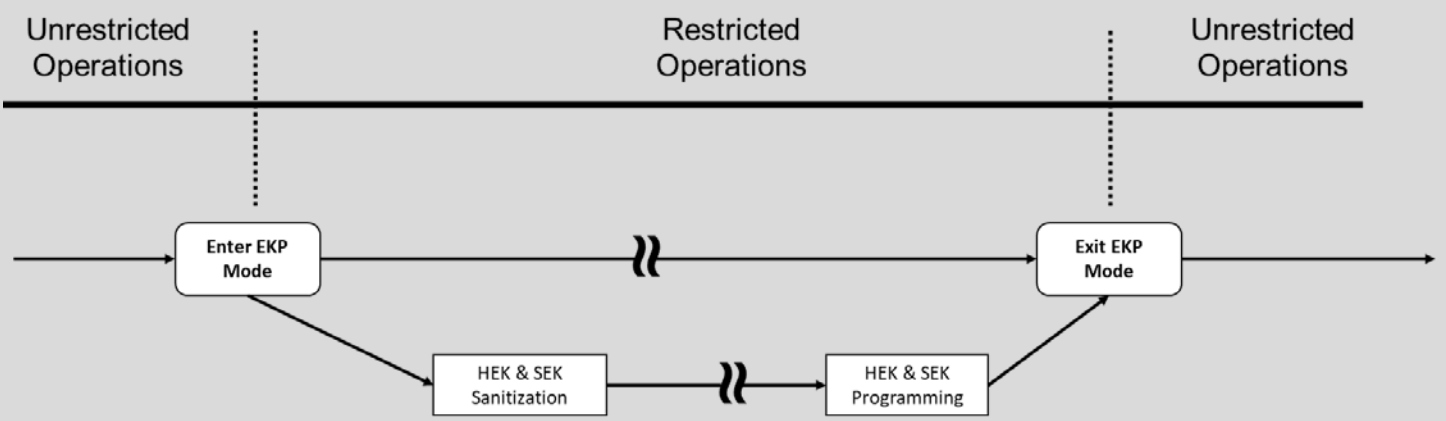
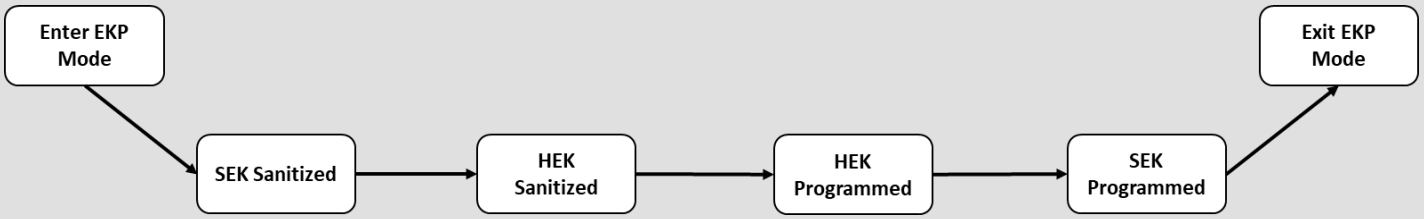


Figure 1 EKP Mode Transitions

The HEK and SEK are used together within the SD and there is an interdependence between these epoch keys. An HEK has to exist (be programmed) before an SEK can be programmed and an SEK has to be sanitized before an HEK can be sanitized. Figure 2 shows the scenario in which both epoch keys are being replaced. Not shown in Figure 2, the SEK Sanitized activity includes sanitizing/eliminating all MEKs that were in use within the SD. In addition, the Exit EKP Mode includes activities to establish one or more new MEKs (cryptographically bound to the epoch keys) to ensure the SD is usable and able to sanitize any user data written to the SD.



 © TCG 2026

Figure 2 Typical Epoch Key State Transitions

The typical operations performed in this EKP mode are shown in

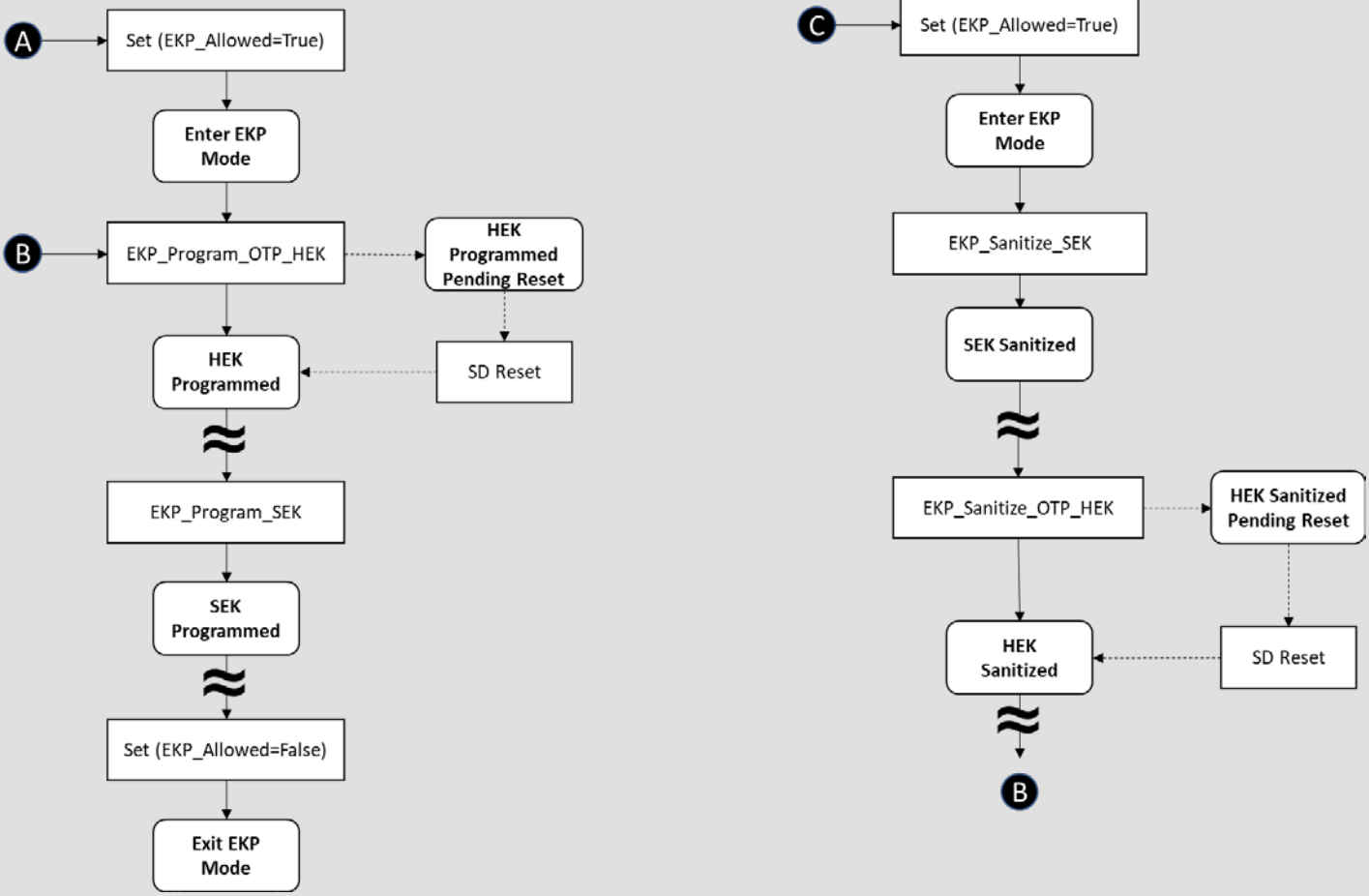


Figure 3, which shows the steps for setting the epoch keys (starting with label A in

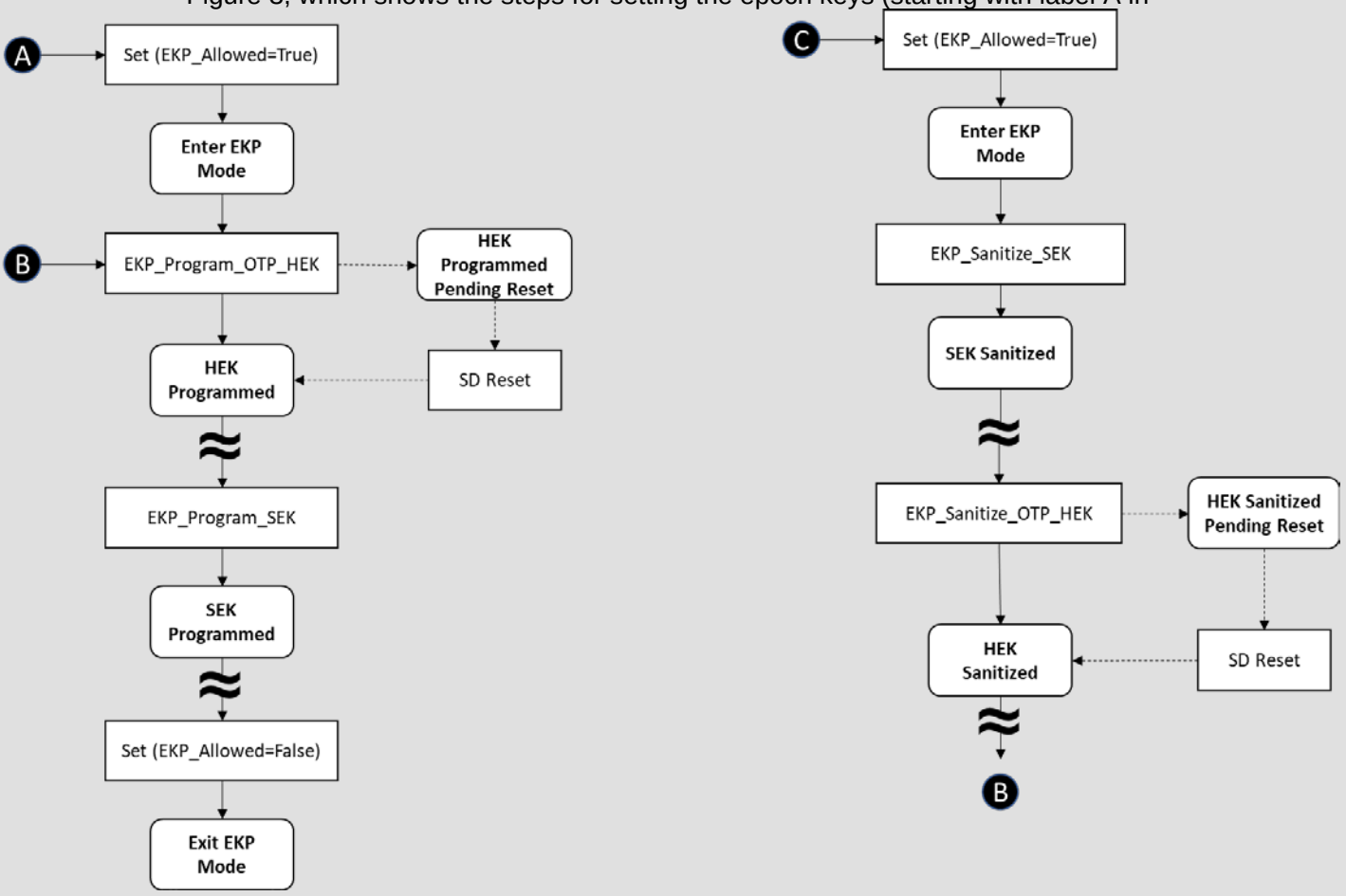


Figure 3) as well as the steps for replacing the epoch keys (starting with label C in Figure 3).

The number of OTP slots usable in an SD for HEKs is expected to be relatively small (e.g., 8 to 16). Thus, these hardware-based sanitizations are likely to be infrequent (e.g., transferring the SD to a new owner). Sanitizing SEKs, on the other hand, is likely to occur on a more frequent basis and there are fewer limitations than the HEK under standard usage.

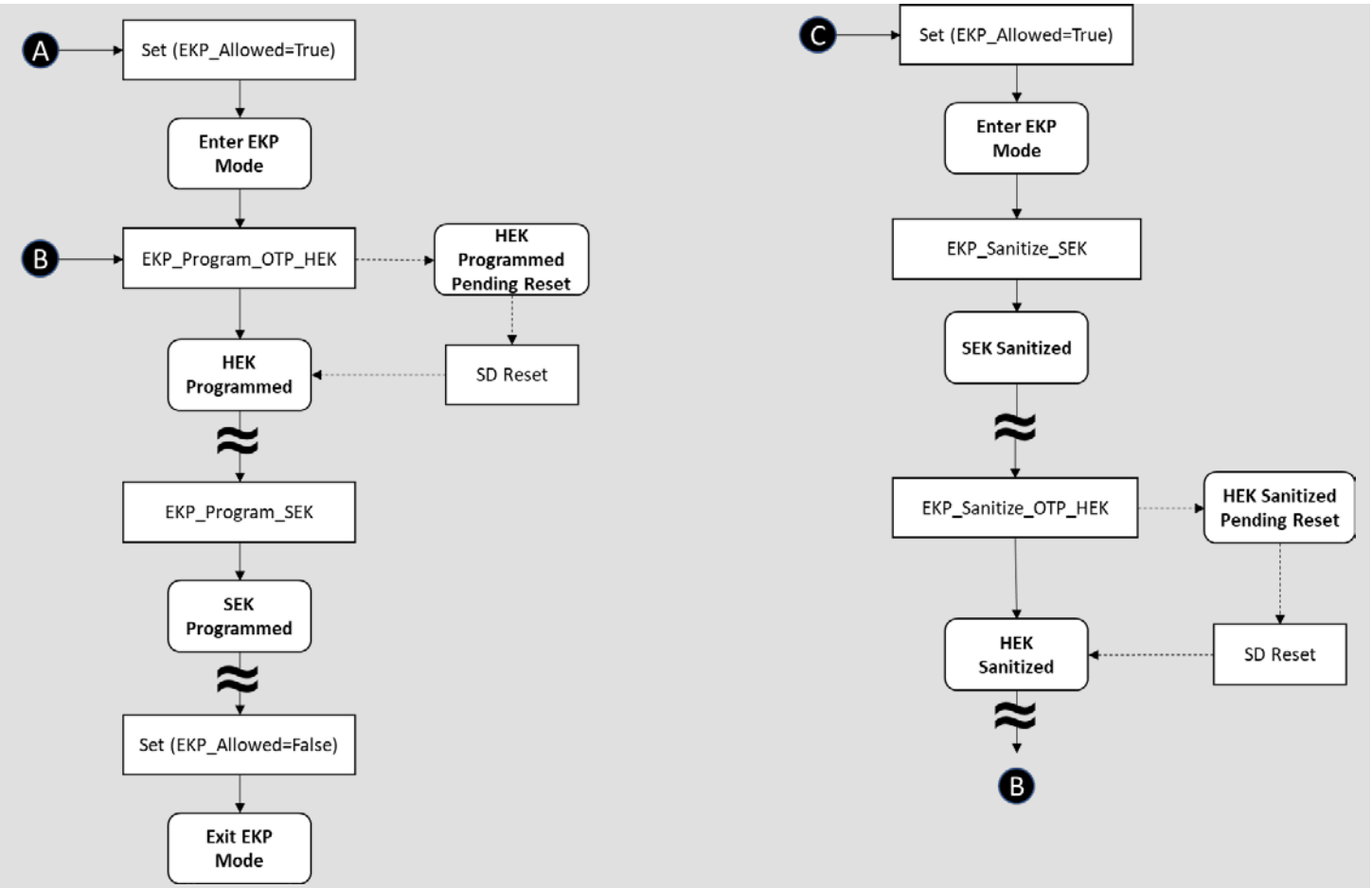


Figure 3 Typical EKP Mode Operations

If an SD consumes all of the OTP HEK slots, there are provisions for addressing this situation by allowing a host to transition the SD to using a permanent HEK (see 4.2.1.3). Figure 4 shows the typical approach to cause the SD to transition to using a permanent HEK.

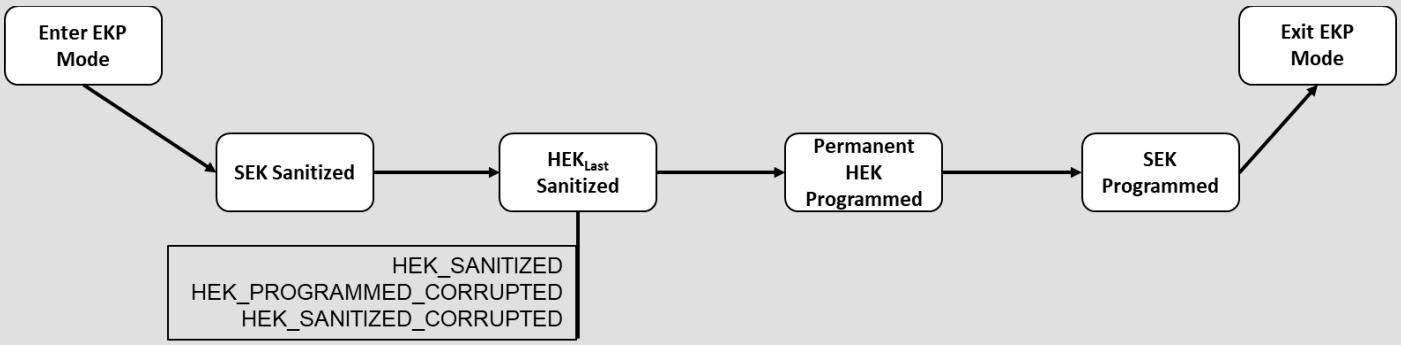


Figure 4 Permanent HEK

SDs that support the EKP feature set are expected to have the initial HEK and SEK programmed as part of the manufacturing process, thereby ensuring the EKP feature set is actively protecting user data without the need for further configuration prior to use of the SD. If the epoch keys are not programmed as part of manufacturing, then they can be programmed at a later time at which point the EKP feature set is actively protecting user data; in such a

scenario, data written to the SD prior to programming the epoch keys cannot be sanitized using this feature set. Also note that if an SD without programmed HEK and SEK transitions into the EKP mode, the SD cannot transition out of the EKP mode until both the HEK and SEK are programmed.

A host can ascertain the state of the EKP feature set by:

- performing a Level 0 discovery, which identifies whether the EKP feature set is supported or not and, if so, the maximum number of HEK sanitizations supported; and
- either using the `EKP_Attested_Report` method or the `Get` method (on the HEK_State and SEK_State columns of the `EpochKeyPurge` object) to determine the state of the epoch keys, specifically whether they are programmed or not

Since the HEK is stored using OTP bits (i.e., hardware) there is a possibility the HEK programming (4.2.1.1) and sanitization (4.2.1.2) operations encounter problems. In such a situation, the HEK status indicates corruption.

For failed HEK programming, a host can retry (one or more times) programming the corrupted OTP slot with the HEK (see label A in Figure 5); if the retry is successful, then the state changes to programmed, otherwise it remains corrupted. Alternatively, the host can abandon attempts to program the current OTP slot and proceed with programming the HEK into the next OTP slot; the HEK history shows that the OTP slot was skipped and marked as corrupted during programming.

Failed HEK sanitization is similar to programming in that the HEK sanitization can be retried one or more times with success resulting in the status changing to sanitized and failure leaving the state as corrupted (see label B in Figure 5). The host can also abandon attempts to sanitize the corrupted OTP slot and proceed with programming the next HEK; the HEK history shows that the OTP slot was skipped and marked as corrupted during sanitization (i.e., the cryptographic erase operation is questionable).

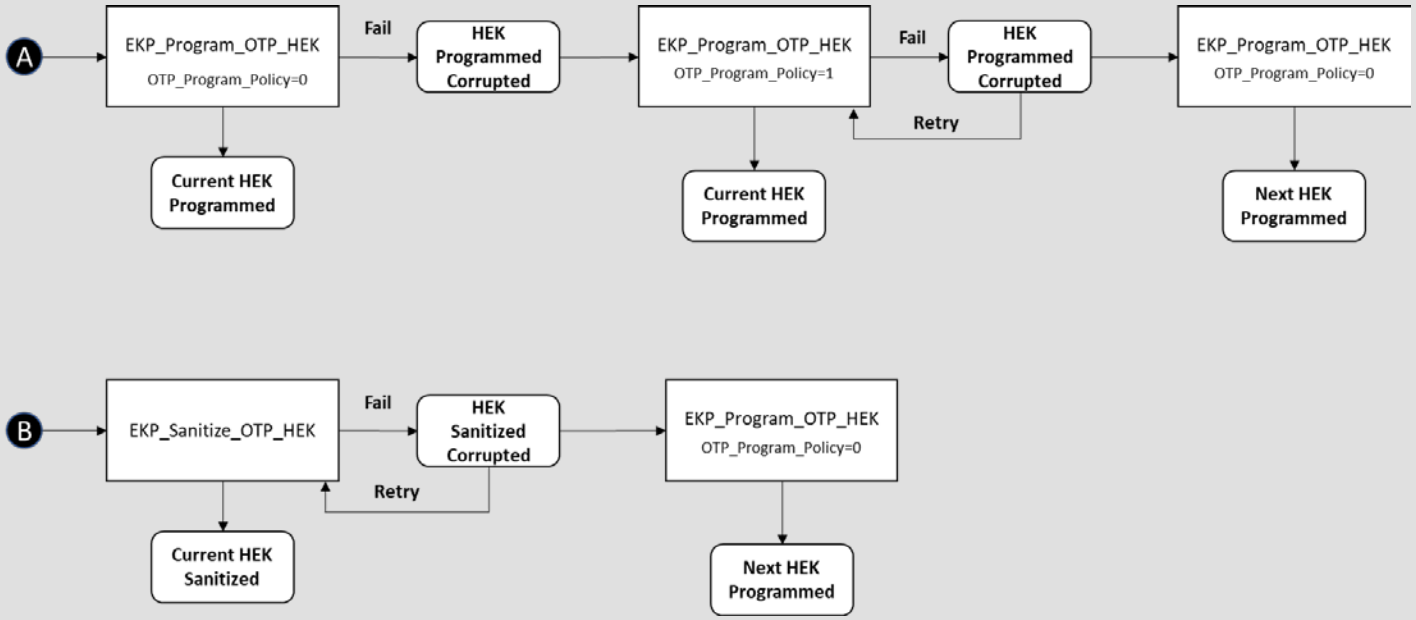


Figure 5 Corrupted HEK Scenarios

The use of OTP bits for the HEKs can result in a situation where the SD has to be reset before OTP bits are accessed and used by the SD. The SD Reset Policy for OTP HEK field in the Level 0 discovery (5.2) response provides information to the host on the need to perform SD resets after changes are made to the OTP bits. The SD Reset Policy for OTP HEK field in the Level 0 discovery applies to both HEK programming (`EKP_Program_OTP_HEK`) and HEK sanitization (`EKP_Sanitize_OTP_HEK`).

When SD resets are necessary, the EKP_Program_OTP_HEK method sets the HEK_State to a value of HEK_PROGRAMMED_PENDING_RESET (i.e., the new HEK was programmed, but the SD is not using it yet). Likewise, the EKP_Sanitize_OTP_HEK method sets the HEK_State to a value of HEK_SANITIZED_PENDING_RESET (i.e., the current HEK has been sanitized, but the SD has not fully recognized it). While a reset of the SD is necessary to clear these pending states, it is important to note that the affected OTP slot had been successfully modified. Also, the EKP mode cannot be exited with these pending reset states, which means the SD reset has to occur while the SD is in the EKP mode.

Figure 6 shows two scenarios following HEK programming that results in a HEK_PROGRAMMED_PENDING_RESET state. The bottom path shows the SD being reset and clearing the pending state before proceeding with the SEK programming. The upper path shows programming the SEK while the HEK is still in a pending state, followed by a reset that clears the HEK pending state. This flexibility can be useful in situations where a host is burning through multiple OTP slots in a single session (e.g., rapidly transitioning to a permanent HEK) without doing a reset after each HEK programming/sanitization method invocation.

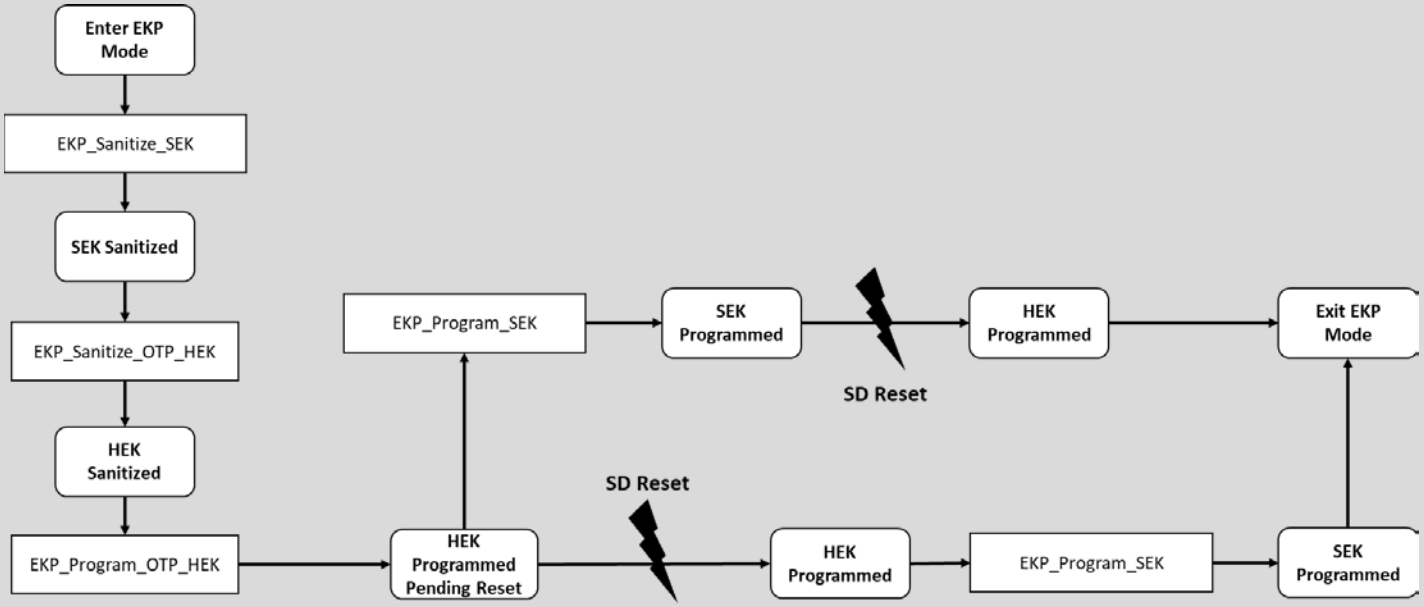


Figure 6 HEK Operations Dependent on Resets

End of Informative Comment

3.4 Attestable EKP Operations

Start of Informative Comment

The EKP feature set permanently maintains (i.e., not affected by SD resets or SSC operations) important information that can be accessed by a host. This information includes:

- Current state of the EKP mode (4.3.1.1.5), which controls the ability to use most EKP methods
- The policy for terminating pending commands/methods when transitioning into the EKP mode
- Current state of the epoch keys (both HEK and SEK)

- The maximum number of usable OTP slots possible for the SD as well as the number of remaining OTP slots available for use
- The state history for all the OTP slots (i.e., previously used, current, and unused)

In situations where an attestable record of the EKP information is desired, the EKP feature set includes the `EKP_Attested_Report` method (4.2.1.6). This method can be invoked at any time on an SD that supports the EKP feature set.

To generate an attested EKP evidence report, the host starts by identifying an acceptable signing certificate that is currently resident in the SD. To do this, the host can access the available certificates by using the `Get` method on the EKP_Cert_List column of the `EKPCertList` object. Each uidref contained in this list provides access to an entity certification, which can be inspected for the algorithms used, ascertain whether it is usable for signing, and verification of the entire certificate chain. The host is expected to select a certificate that is appropriate to its needs and to use the uidref in future invocations of the `EKP_Attested_Report` method. It is possible that the certificates that are resident within the SD can change over time, so the recommended practice to repeat this selection process when security requirements change (e.g., transitions to quantum resistant cryptography) or there are significant gaps in time between report generations.

Once the host has identified a usable certificate, the `EKP_Attested_Report` method can be invoked. As part of this invocation, the host provides a nonce, specifies the report type (COSE protected CBOR-encoded byte string is the only current option), and the uidref of the certificate.

The following are important aspects of the EKP_Attested_Report method:

- The current states and other important details are bundled in a tagged format as evidence claims
- To determine “freshness”, a host specifies a nonce as a parameter and this nonce is included in the returned signed EKP evidence.
- The returned EKP evidence is signed by a host-selected certificate.
- The host has the ability to select the format of the returned EKP evidence.

A host can retrieve EKP evidence information by using the `Get` method on columns in the `EpochKeyPurge` object (4.3.1.1). A typical use is to check the current state of one or more columns prior to invocation of an EKP method to verify the particular EKP method can be invoked successfully or to verify an EKP method (via another `Get` method invocation after the EKP method completes).

Currently, the EKP evidence is formatted into an object based on the Concise Binary Object Representation (CBOR), which is a data format that serializes structured data into a compact binary form. CBOR Object Signing and Encryption (COSE) builds on this to include signatures, message authentication codes (MACs) and encryption; signing is the only COSE capability used by this feature set. The EKP evidence payload is created from CBOR-encoded data elements and then protected with a cryptographic signature using COSE.

End of Informative Comment

4 Security Subsystem Class (SSC) Specific Functionality

This section specifies the additional SSC specific functionality required to support the Epoch Key Purge (EKP) Feature Set.

4.1 General

This section defines general requirements for the EKP feature set.

An SD including the EKP feature set capabilities SHALL have hardware-based [4][5][6][7][8], symmetric key encryption capabilities that are always in use for protection of user data.

An SD including the EKP feature set capabilities SHALL use a pair of key-derivation keys [9] known as epoch keys that are used jointly (cryptographically combined) for the purpose of storage sanitization, based on the purge technique of cryptographic erase. These two keys SHALL be a Hard Epoch Key (HEK) and a Soft Epoch Key (SEK) and only one of each SHALL be active at any point in time.

All media encryption keys (symmetric data-encryption keys [9]) used by the SD as part of its data at rest encryption capability SHALL be cryptographically bound to the epoch keys.

The SD SHALL store HEKs in secure hardware-based, one-time programmable (OTP) bits (e.g., fuses) that are combined into OTP slots.

The OTP bits used by the SD for the HEK SHALL be subject to key sanitization, as described in ISO/IEC 19790 [18] (e.g., overwriting with all zeros, all ones, or random data).

The SD SHALL store SEKs in secure, non-volatile media.

Sanitization of the HEK and/or SEK by the SD SHALL render stored user data unrecoverable (i.e., the SD performs cryptographic erase as a purge sanitization technique [9]).

The SD SHALL have at least one digital certificate available for signing purposes and this certificate can be used to sign EKP evidence returned to a host.

The SD SHALL NOT be able to disable the EKP feature set and the epoch keys and EKP metadata (see 4.3.1.1) SHALL be persistent throughout the life of the SD.

When the SD has transitioned into the EKP mode, the SD SHALL restrict permitted operations to those that are permitted during user data removal methods [3] and any attempted operation that is not permitted in the EKP mode SHALL result in a failure with a status of Data Protection Error.

4.2 Methods

Start of Informative Comment

The EKP Feature Set uses multiple new methods to control the OTP functionality of an SD. These methods program and sanitize the epoch keys (i.e., HEK and SEK). These epoch keys are used by the SD to protect sensitive information through data at-rest encryption as well as media sanitization in the form a cryptographic erase (purge sanitization technique) [9].

With the exception of the method to report epoch states (`EKP_Attested_Report`), invocation of all these methods requires the SD to be in a special EKP mode, which is accomplished with the `Set` method.

It is important to note that when the SD is in the EKP mode, there are restrictions on what the SD can do (e.g., read and write operations are disabled). In addition, significant amounts of time can elapse in between the invocation of the EKP methods.

End of Informative Comment

This section defines new methods and modifications to existing methods required for this feature set.

4.2.1 New Methods

This section defines the new methods that are required to support this feature set.

The new methods defined are Admin Template-specific methods.

4.2.1.1 EKP_Program_OTP_HEK

Start of Informative Comment

The EKP_Program_OTP_HEK method programs a random HEK into an OTP slot. Failure to meet the following preconditions results in the method failing:

- The SD is in a state that permits use of the EKP methods (EKP_Allowed = True); SD read and write operations are blocked
- An SEK is not currently programmed (i.e., unused or sanitized)
- An HEK is not currently programmed (i.e., unused, sanitized, sanitized pending reset, sanitized corrupted, or programmed corrupted)
- There is an available OTP slot that can be programmed (Remaining_HEK_Sanitizations > 0)

The method includes an input parameter that under typical usage (OTP_Program_Policy = 0) indicates the next available HEK slot is to be used for programming the HEK. However, if the HEK state indicates a previous program corruption, this method can be used with a special input parameter (OTP_Program_Policy = 1) to retry programming the HEK into the current OTP slot.

If the OTP slot has been successfully programmed, then the HEK state is changed to HEK_PROGRAMMED, the method succeeds, and returns a status of SUCCESS. The HEK history state is also updated and the Remaining_HEK_Sanitizations is decremented by 1.

The process to program an OTP slot can take time and be subjected to interruptions (e.g., reset of the SD) or issues associate with OTP technology used by the SD. As such, the SD is expected to change the HEK state to “programmed corrupted” prior to making any changes to the OTP slot; this is particularly important to allow for recoveries after SD resets or power interrupts (i.e., the SD cannot report a method failure). If the method fails, the status will be set to FAIL. To determine the exact reason for a failure, the EpochKeyPurge object data can be inspected.

A good practice is to always check the HEK state after invoking this method to confirm the SD is in the expected EKP state/configuration.

This method can be used to address the following HEK corruption scenarios:

- Program corruption: The method can attempt to program the current “programmed corrupted” OTP slot by invoking the method with the OTP_Program_Policy input parameter set to 1. If the corruption persists after subsequent attempts to program the current OTP slot, the method can be used to proceed with using the next available OTP slot by invoking the method with the OTP_Program_Policy input parameter set to 0. In this latter situation, the HEK history state for the current slot will remain PROGRAMMED_CORRUPTED, the Remaining_HEK_Sanitizations is decremented by 1, and the invocation of the method proceeds as normal (success or failure).
- Sanitization corruption: The method can be used to program an HEK into the next OTP slot, but have no effect on the current OTP slot; the method can only be invoked with the OTP_Program_Policy input parameter set to 0 when addressing a sanitize corruption. In this situation, the HEK history state for the current slot will remain SANITIZED_CORRUPTED, the Remaining_HEK_Sanitizations is decremented by 1, and the invocation of the method proceeds as normal (success or failure).

When an HEK is programmed into an OTP slot, the next logical step is to program the SEK.

The HEK_State_List is also updated to reflect the outcome of this method. The list entry is set to PROGRAMMED when the method is successful and PROGRAMMED_CORRUPTED when the OTP bits are not completely programmed.

End of Informative Comment

The EKP_Program_OTP_HEK method is an Admin Template-specific method.

The following pseudo-code is the signature of the EKP_Program_OTP_HEK method (see 4.2.1.1.3 for more information).

```
EpochKeyPurgeObjectUID.EKP_Program_OTP_HEK [
    OTP_Program_Policy : uinteger]
=>
[ ]
```

Method UID: 00 00 00 06 00 00 0C 01

4.2.1.1.1 Parameter Descriptions

4.2.1.1.1.1 OTP_Program_Policy

The OTP_Program_Policy parameter specifies a policy value (see Table 2) that determines whether an HEK is to be programmed in the current HEK OTP slot or the next slot.

Table 2 OTP_Program_Policy Table Description

Value	Description
0x0000	Use the next HEK OTP slot to program the HEK.
0x0001	Use the current HEK OTP slot to program the HEK.
0x0002 to 0xFFFF	Reserved

4.2.1.1.2 Returned Value Descriptions

The EKP_Program_OTP_HEK method returns no values.

4.2.1.1.3 EKP_Program_OTP_HEK Method Operation

Start of Informative Comment

Check to see if the SD is in a state that permits the HEK to be operated upon.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is set to a value of False, then the EKP_Program_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

The EKP_Program_OTP_HEK method is not usable within a transaction.

End of Informative Comment

If EKP_Program_OTP_HEK method is invoked within a transaction, then the EKP_Program_OTP_HEK method SHALL fail with a status of TRANSACTION_FAILURE.

Start of Informative Comment

Typically, an HEK is programmed after the previous HEK and SEK were sanitized. If, however, the current HEK has been corrupted due to a failure or issue while programming or sanitizing the HEK, this method can allow the host to attempt to recover by programming the HEK. A less common situation wherein the manufacturer did not program the initial epoch keys could result in an HEK state of HEK_UNUSED.

End of Informative Comment

If:

a) the HEK_State column of the EpochKeyPurge object is set to a value other than HEK_SANITIZED, HEK_SANITIZED_PENDING_RESET, HEK_PROGRAMMED_CORRUPTED, HEK_UNUSED, or HEK_SANITIZED_CORRUPTED; or

b) the SEK_State column of the EpochKeyPurge object is set to a value other than SEK_UNUSED or SEK_SANITIZED,

then the EKP_Program_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

Check to see if the SD is using a permanent HEK. Permanent HEK cannot be programmed.

End of Informative Comment

If the HEK_State column of the EpochKeyPurge object is set to a value of HEK_PERMANENT, then the EKP_Program_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

Check to make sure that the OTP slot that has not been programmed previously, or if it was corrupted as result of a previous attempt to program an HEK into an OTP.

When the host is addressing a previously failed programming attempt, the options are to retry or to proceed with programming a new OTP slot. The input parameter (OTP_Program_Policy) determines the option.

End of Informative Comment

If the value of OTP_Program_Policy is not equal to 0 or 1, then the EKP_Program_OTP_HEK method SHALL fail with a status of INVALID_PARAMETER.

If:

a) the value of OTP_Program_Policy is equal to 0 and the HEK_State column of the EpochKeyPurge object is not set to HEK_SANITIZED, HEK_SANITIZED_PENDING_RESET, HEK_PROGRAMMED_CORRUPTED, HEK_SANITIZED_CORRUPTED, or HEK_UNUSED; or

b) the value of OTP_Program_Policy is equal to 1 and the HEK_State column of the EpochKeyPurge object is not set to HEK_PROGRAMMED_CORRUPTED,

then the EKP_Program_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

With the pre-conditions being met, proceed with the EKP_Program_OTP_HEK method to program the OTP bits for the next HEK. It is possible that the programming of the HEK could fail; account for resets during the programming by setting the state to indicate corruption prior to programming the HEK. Both a programmed and corrupted programming operation result in the consumption of an OTP slot.

End of Informative Comment

If:

a) the EKP_Program_OTP_HEK method is invoked,

b) the value of the EKP_Allowed column of the EpochKeyPurge object is True;

c) the value of OTP_Program_Policy is equal to

1. 0 and the HEK_State column of the EpochKeyPurge object is set to HEK_SANITIZED, HEK_SANITIZED_PENDING_RESET, HEK_PROGRAMMED_CORRUPTED, HEK_SANITIZED_CORRUPTED, or HEK_UNUSED; or
2. 1 and the HEK_State column of the EpochKeyPurge object is set to HEK_PROGRAMMED_CORRUPTED;

d) the value of the SEK_State column of the EpochKeyPurge object is SEK_UNUSED or SEK_SANITIZED; and

e) the value of the Remaining_HEK_Sanitizations of the EpochKeyPurge object is greater than 0,

then the following changes SHALL be made:

1) The value of the HEK_State column of the EpochKeyPurge object SHALL be set to HEK_PROGRAMMED_CORRUPTED.

2) If the value of OTP_Program_Policy is equal to 0, then the value of the Remaining_HEK_Sanitizations of the EpochKeyPurge object SHALL be decremented by 1.

3) The value of the HEK_State_List[MaxSupported_HEK_Sanitizations - Remaining_HEK_Sanitizations] SHALL be set to PROGRAMMED_CORRUPTED.

4) The HEK SHALL be programmed into the OTP bits.

5) If all the OTP bits for the HEK were successfully programed with new key material and the SD does not need to be reset, then:

1. The value of the HEK_State column of the EpochKeyPurge object SHALL be set to a value of HEK_PROGRAMMED;
2. The value of the HEK_State_List[MaxSupported_HEK_Sanitizations - Remaining_HEK_Sanitizations] SHALL be set to PROGRAMMED; and
3. The EKP_Program_OTP_HEK method SHALL succeed with a return status of SUCCESS.

6) If all the OTP bits for the HEK were successfully programed with new key material and the SD does need to be reset, then:

1. The value of the HEK_State column of the EpochKeyPurge object SHALL be set to a value of HEK_PROGRAMMED_PENDING_RESET;
2. The value of the HEK_State_List[MaxSupported_HEK_Sanitizations - Remaining_HEK_Sanitizations] SHALL be set to PROGRAMMED; and
3. The EKP_Program_OTP_HEK method SHALL succeed with a return status of SUCCESS.

7) If all the OTP bits for the HEK were not successfully programed, then the EKP_Program_OTP_HEK method SHALL fail with a status of FAIL.

4.2.1.2 EKP_Sanitize_OTP_HEK

Start of Informative Comment

The EKP_Sanitize_OTP_HEK method sanitizes, as described in ISO/IEC 19790 [18] (e.g., overwriting with all zeros, all ones, or random data), the current HEK OTP slot. Failure to meet the following preconditions results in the method failing:

- The SD is in a state that permits use of the EKP methods (EKP_Allowed = True); read and write operations are blocked
- An SEK is not currently programmed
- An HEK is currently programmed (or programmed, pending reset)
- A previous attempt to sanitize an HEK had not completed successfully and resulted in a corruption

If the OTP slot has been successfully sanitized, then the HEK state is changed to HEK_SANITIZED, the method succeeds, and returns a status of SUCCESS.

The process to sanitize an OTP slot can take time and be subjected to interruptions (e.g., reset of the SD) or issues associated with OTP technology used by the SD. As such, the SD is expected to change the HEK state to HEK_ SANITIZED _CORRUPTED prior to making any changes to the OTP slot; as with programming the HEK, this is particularly important to allow for recoveries after SD resets or power interrupts (i.e., the SD cannot report a method failure). If the method fails, the status will be set to FAIL. To determine the exact reason for a failure, the EpochKeyPurge object data can be inspected.

A good practice is to always check the HEK state after invoking this method to confirm the SD is in the expected EKP state/configuration.

This method can be used to address the following HEK corruption scenario:

- Sanitization corruption: The method can attempt to sanitize the current OTP slot with a state of HEK_ SANITIZED _CORRUPTED. If the corruption persists after one or more subsequent attempts to sanitize the current OTP slot, then the only alternative is to program the next available HEK OTP slot or, if no HEK OTP slot is available, transition to a permanent HEK.

When an HEK OTP slot has been sanitized, the next logical step is to program the next HEK or transition to a permanent HEK, if all the OTP slots have been consumed.

The HEK_State_List is also updated to reflect the outcome of this method. The list entry is set to SANITIZED when the method is successful and SANITIZED_CORRUPTED when the OTP bits are not completely sanitized.

End of Informative Comment

The EKP_Sanitize_OTP_HEK method is an Admin Template-specific method.

The following pseudo-code is the signature of the EKP_Sanitize_OTP_HEK method (see 4.2.1.2.3 for more information).

```
EpochKeyPurgeObjectUID.EKP_Sanitize_OTP_HEK [ ]
=>
[ ]
```

Method UID: 00 00 00 06 00 00 0C 02

 © TCG 2026

4.2.1.2.1 Parameter Descriptions

The EKP_Sanitize_OTP_HEK method has no parameters.

4.2.1.2.2 Returned Value Descriptions

The EKP_Sanitize_OTP_HEK method returns no values.

4.2.1.2.3 EKP_Sanitize_OTP_HEK Method Operation

Start of Informative Comment

Check to see if the SD is in a state that permits the HEK to be operated upon.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is set to a value of False, then the EKP_Sanitize_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

The EKP_Sanitize_OTP_HEK method is not usable within a transaction.

End of Informative Comment

If EKP_Sanitize_OTP_HEK method is invoked within a transaction, then the EKP_Sanitize_OTP_HEK method SHALL fail with a status of TRANSACTION_FAILURE.

Start of Informative Comment

Check to see if the SD is using a permanent HEK. Permanent HEK cannot be sanitized.

End of Informative Comment

If the HEK_State column of the EpochKeyPurge object is set to a value of HEK_PERMANENT, then the EKP_Sanitize_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

An HEK can only be sanitized when an HEK has been programmed and after the SEK was sanitized. If the current HEK has been corrupted due to a failure or issue while sanitizing the HEK, allow the host to attempt to recover by sanitizing the HEK again. Attempting to sanitize the HEK after it was successfully sanitized is not permitted and will result in a method failure; the host can check the HEK_State after receiving such a failure to determine whether the HEK was sanitized.

A less common situation wherein the manufacturer did not program the initial epoch keys could result in an SEK state of SEK_UNUSED, which is treated as if the SEK was sanitized.

End of Informative Comment

If:

a) the HEK_State column of the EpochKeyPurge object is set to a value other than HEK_PROGRAMMED, HEK_PROGRAMMED_PENDING_RESET, or HEK_SANITIZED_CORRUPTED; or

b) the SEK_State column of the EpochKeyPurge object is set to a value other than SEK_UNUSED or SEK_SANITIZED,

then the EKP_Sanitize_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

With the pre-conditions being met, proceed with the EKP_Sanitize_OTP_HEK method to sanitize the OTP bits the current HEK. Check to make sure no problems were encountered during HEK sanitization of the OTP slot. It is possible

that the sanitization of the HEK could fail (e.g., power disruption); to account for resets during the sanitizing, set the state to indicate corruption prior to sanitizing the HEK.

Sanitizing the HEK is the final step in the EKP sanitization process; while it does render all data on the SD sanitized, this is of minimal consequence as previous sanitization of the SEK is sufficient to sanitize the SD. This method, however, provides the attestable sanitization, which is achieved when the current HEK is sanitized.

End of Informative Comment

If:

a) the EKP_Sanitize_OTP_HEK method is invoked;

b) the value of the EKP_Allowed column of the EpochKeyPurge object is True;

c) the value of the HEK_State column of the EpochKeyPurge object is HEK_PROGRAMMED, HEK_PROGRAMMED_PENDING_RESET, or HEK_SANITIZED_CORRUPTED; and

d) the value of the SEK_State column of the EpochKeyPurge object is SEK_SANITIZED,

then the following changes SHALL be made:

1) The value of the HEK_State column of the EpochKeyPurge object SHALL be set to HEK_SANITIZED_CORRUPTED.

2) The value of the HEK_State_List[MaxSupported_HEK_Sanitizations-Remaining_HEK_Sanitizations] SHALL be set to SANITIZED_CORRUPTED.

3) The OTP bits for the current HEK SHALL be sanitized.

4) If the OTP bits for the current HEK were successfully sanitized and the SD does not need to be reset, then

   1. the value of the HEK_State column of the EpochKeyPurge object SHALL be set to HEK_SANITIZED;

   2. The value of the HEK_State_List[MaxSupported_HEK_Sanitizations-Remaining_HEK_Sanitizations] SHALL be set to SANITIZED; and

   3. the EKP_Sanitize_OTP_HEK method SHALL succeed with a return status of SUCCESS.

5) If the OTP bits for the current HEK were successfully sanitized and the SD does need to be reset, then

   1. The value of the HEK_State column of the EpochKeyPurge object SHALL be set to a value of HEK_SANITIZED_PENDING_RESET;

   2. The value of the HEK_State_List[MaxSupported_HEK_Sanitizations-Remaining_HEK_Sanitizations] SHALL be set to SANITIZED; and

   3. the EKP_Sanitize_OTP_HEK method SHALL succeed with a return status of SUCCESS.

6) If the OTP bits for the current HEK were not successfully sanitized, then

   1. the EKP_Sanitize_OTP_HEK method SHALL fail with a return status of FAIL.

4.2.1.3 EKP_Enable_Permanent_HEK

Start of Informative Comment

Once all the OTP slots for the HEK have been used and the last slot has been sanitized or is in a corrupted state, the SD can transition to the use of a static or permanent HEK. For the purposes of this specification, a “permanent” HEK is an HEK stored in non-volatile storage (e.g., flash) and is implementation-specific; implementations can use an HEK that is a static key or a well-known constant (e.g., all zeroes or all ones). The permanent HEK is not expected to contribute significantly to the security of the SD.

The EKP_Enable_Permanent_HEK method enables an SD to continue to be used after all of the HEK OTP slots have been consumed (i.e., subsequent sanitization is not attestable). Failure to meet the following preconditions results in the method failing:

- The SD is in a state that permits use of the EKP methods (EKP_Allowed = True); read and write operations are blocked
- An SEK is not currently programmed
- An HEK is not currently programmed (i.e., sanitized, sanitized pending reset, sanitized corrupted, or programmed corrupted)
- There are no available HEK OTP slots that can be programmed (Remaining_HEK_Sanitizations = 0)

If the permanent HEK has been enabled/set, then the HEK state is changed to HEK_PERMANENT, the method succeeds, and returns a status of SUCCESS. The next logical step is to program the SEK.

If the SD is unable to transition to using a permanent HEK, the SD is rendered unusable because read and write operation remain blocked.

End of Informative Comment

The EKP_Enable_Permanent_HEK method is an Admin Template-specific method.

The following pseudo-code is the signature of the EKP_Enable_Permanent_HEK method (see 4.2.1.3.3 for more information).

```
EpochKeyPurgeObjectUID.EKP_Enable_Permanent_HEK [ ]
=>
[ ]
```

Method UID: 00 00 00 06 00 00 0C 03

4.2.1.3.1 Parameter Descriptions

The EKP_Enable_Permanent_HEK method has no parameters.

4.2.1.3.2 Returned Value Descriptions

The EKP_Enable_Permanent_HEK method returns no values.

4.2.1.3.3 EKP_Enable_Permanent_HEK Method Operation

Start of Informative Comment

Check to see if the SD is in a state that permits the HEK to be operated upon.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is set to a value of False, then the EKP_Enable_Permanent_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

The EKP_Enable_Permanent_HEK method is not usable within a transaction.

End of Informative Comment

If EKP_Enable_Permanent_HEK method is invoked within a transaction, then the EKP_Enable_Permanent_HEK method SHALL fail with a status of TRANSACTION_FAILURE.

Start of Informative Comment

Check to see if the SD is using a permanent HEK.

End of Informative Comment

If the HEK_State column of the EpochKeyPurge object is set to a value of HEK_PERMANENT, then the EKP_Sanitize_OTP_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

An HEK can only be permanently programmed after the previous HEK and SEK were sanitized and when there are no more OTP slots available.

End of Informative Comment

If:

a) the HEK_State column of the EpochKeyPurge object is set to a value other than HEK_SANITIZED, HEK_SANITIZED_PENDING_RESET, HEK_PROGRAMMED_CORRUPTED, or HEK_SANITIZED_CORRUPTED;

b) the SEK_State column of the EpochKeyPurge object is set to a value other than SEK_SANITIZED; or

c) the value of the Remaining_HEK_Sanitizations of the EpochKeyPurge object is greater than 0,

then the EKP_Enable_Permanent_HEK method SHALL fail with a status of FAIL.

Start of Informative Comment

Program the permanent HEK and change the HEK_State to HEK_PERMANENT.

End of Informative Comment

If:

a) the EKP_Enable_Permanent_HEK method is invoked;

b) the value of the EKP_Allowed column of the EpochKeyPurge object is True;

c) the value of the HEK_State column of the EpochKeyPurge object is set to HEK_SANITIZED, HEK_SANITIZED_PENDING_RESET, HEK_PROGRAMMED_CORRUPTED, or HEK_SANITIZED_CORRUPTED;

d) the value of the SEK_State column of the EpochKeyPurge object is SEK_SANITIZED; and

e) the value of the Remaining_HEK_Sanitizations of the EpochKeyPurge object is equal to 0,

then the following changes SHALL be made:

1) The permanent HEK SHALL be programmed.

2) If the permanent HEK was successfully programmed, then

   1. The HEK_State column of the EpochKeyPurge object SHALL be set to a value of HEK_PERMANENT.

   2. The EKP_Enable_Permanent_HEK method SHALL succeed with a return status of SUCCESS.

3) If the permanent HEK was not successfully programmed, then the EKP_Enable_Permanent_HEK method SHALL fail with a return status of FAIL.

 © TCG 2026

4.2.1.4 EKP_Program_SEK

Start of Informative Comment

The EKP_Program_SEK method programs a random SEK into a non-volatile storage location (e.g., flash); SEK programming is an atomic operation. Failure to meet the following preconditions results in the method failing:

- The SD is in a state that permits use of the EKP methods (EKP_Allowed = True); read and write operations are blocked
- An SEK is not currently programmed
- An HEK is currently programmed

If the SEK has been successfully programmed, then the SEK state is changed to SEK_PROGRAMMED, the method succeeds, and returns a status of SUCCESS. The next logical step is for the SD to exit the state that permits the use of EKP methods (EKP_Allowed = False), which can be accomplished with the Set method (4.2.2.1.1.2).

End of Informative Comment

The EKP_Program_SEK method is an Admin Template-specific method.

The following pseudo-code is the signature of the EKP_Program_SEK method (see 4.2.1.4.3 for more information).

```
EpochKeyPurgeObjectUID. EKP_Program_SEK [ ]
=>
[ ]
```

Method UID: 00 00 00 06 00 00 0C 04

4.2.1.4.1 Parameter Descriptions

The EKP_Program_SEK method has no parameters.

4.2.1.4.2 Returned Value Descriptions

The EKP_Program_SEK method returns no values.

4.2.1.4.3 EKP_Program_SEK Method Operation

Start of Informative Comment

Check to see if the SD is in a state that permits the SEK to be operated upon.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is set to a value of False, then the EKP_Program_SEK method SHALL fail with a status of FAIL.

Start of Informative Comment

The EKP_Program_SEK method is not usable within a transaction.

End of Informative Comment

If EKP_Program_SEK method is invoked within a transaction, then the EKP_Program_SEK method SHALL fail with a status of TRANSACTION_FAILURE.

Start of Informative Comment

Check to make sure the HEK is already programmed and the SEK is already sanitized as a prerequisite for programming the SEK.

End of Informative Comment

If:

a) HEK_State column of the EpochKeyPurge object is set to a value other than HEK_PROGRAMMED, HEK_PROGRAMMED_PENDING_RESET, or HEK_PERMANENT; or

b) SEK_State column of the EpochKeyPurge object is set to a value other than SEK_UNUSED or SEK_SANITIZED,

then the EKP_Program_SEK method SHALL method SHALL fail with a status of FAIL.

Start of Informative Comment

Program the SEK. If the attempt to program the SEK is not successful, do not change the SEK_State and return a method failure.

Once both the HEK and SEK have been programmed, the SD can address the MEKs.

End of Informative Comment

If:

a) the EKP_Program_SEK method is invoked;

b) the value of the EKP_Allowed column of the EpochKeyPurge object is True;

c) the value of the HEK_State column of the EpochKeyPurge object is HEK_PROGRAMMED, HEK_PROGRAMMED_PENDING_RESET, or HEK_PERMANENT; and

d) the value of the SEK_State column of the EpochKeyPurge object is SEK_UNUSED or SEK_SANITIZED,

then the following changes SHALL be made:

1) The SEK SHALL be programmed with new key material.

2) If the SEK was successfully programmed, then

   1. set the SEK_State column of the EpochKeyPurge object to a value of SEK_PROGRAMMED; and

   2. the EKP_Program_SEK method SHALL succeed with a return status of SUCCESS.

3) If the SEK was not successfully programmed, then the EKP_Program_SEK method SHALL fail with a return status of FAIL.

4.2.1.5 EKP_Sanitize_SEK

Start of Informative Comment

The EKP_Sanitize_SEK method sanitizes, as described in ISO/IEC 19790 [18] (e.g., overwriting with all zeros, all ones, or random data), the SEK currently stored in a non-volatile storage location (e.g., flash); SEK sanitization is an atomic operation. Failure to meet the following preconditions results in the method failing:

- The SD is in a state that permits use of the EKP methods (EKP_Allowed = True); read and write operations are blocked
- An SEK is currently programmed
- An HEK is currently programmed (or programmed, pending reset)

If the SEK has been successfully sanitized, then the SEK state is changed to SEK_SANITIZED, the method succeeds, and returns a status of SUCCESS. Sanitization of the SEK is a cryptographic erase operation [9] that causes the immediate sanitization of data (user data removal [3] operation). To ensure immediate enforcement, the SD is expected to unload any cached symmetric data encryption keys (e.g., MEKs) available to the encryption engine.

The next logical step for the SD is one of the following:

- Program the next SEK
- Sanitize the current HEK

End of Informative Comment

The EKP_Sanitize_SEK method is an Admin Template-specific method.

The following pseudo-code is the signature of the EKP_Sanitize_SEK method (see 4.2.1.5.3 for more information).

```
EpochKeyPurgeObjectUID.EKP_Sanitize_SEK [ ]
=>
[ ]
```

Method UID: 00 00 00 06 00 00 0C 05

4.2.1.5.1 Parameter Descriptions

The EKP_Sanitize_SEK method has no parameters.

4.2.1.5.2 Returned Value Descriptions

The EKP_Sanitize_SEK method returns no values.

4.2.1.5.3 EKP_Sanitize_SEK Method Operation

Start of Informative Comment

Check to see if the SD is in a state that permits the SEK to be operated upon.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is set to a value of False, then the EKP_Sanitize_SEK method SHALL fail with a status of FAIL.

Start of Informative Comment

The EKP_Sanitize_SEK method is not usable within a transaction.

End of Informative Comment

If EKP_Sanitize_SEK method is invoked within a transaction, then the EKP_Sanitize_SEK method SHALL fail with a status of TRANSACTION_FAILURE.

Start of Informative Comment

An SEK can only be sanitized when an HEK and SEK have been programmed.

End of Informative Comment

If

a) the HEK_State column of the EpochKeyPurge object is set to a value other than HEK_PROGRAMMED, HEK_PROGRAMMED_PENDING_RESET, or HEK_PERMANENT; or

b) the SEK_State column of the EpochKeyPurge object is set to a value other than SEK_PROGRAMMED,

then the EKP_Sanitize_SEK method SHALL fail with a status of FAIL.

Start of Informative Comment

With the pre-conditions being met, proceed with the EKP_Sanitize_SEK method to sanitize the SEK. Check to make sure no problems were encountered during SEK sanitization; this specification does not identify any error conditions that might exist.

Sanitizing the SEK is the first step in the EKP sanitization process and it does render all data on the SD sanitized. By itself, however, this method is not adequate to provide attestable sanitization, which is achieved when the current SEK is sanitized.

Once the SEK has been sanitized, the SD is not permitted to use any previous MEKs.

End of Informative Comment

If:

a) the EKP_Sanitize_SEK method is invoked;

b) the value of the EKP_Allowed column of the EpochKeyPurge object is True;

c) the value of the HEK_State column of the EpochKeyPurge object is HEK_PROGRAMMED, HEK_PROGRAMMED_PENDING_RESET, or HEK_PERMANENT; and

d) the value of the SEK_State column of the EpochKeyPurge object is SEK_PROGRAMMED,

then the following changes SHALL be made:

1) The existing SEK SHALL be sanitized.

2) If the SEK was successfully sanitized, then

   1. set the SEK_State column of the EpochKeyPurge object to a value of SEK_SANITIZED;

   2. apply key sanitization to any previously loaded or cached MEKs; and

   3. the EKP_Sanitize_SEK method SHALL succeed with a return status of SUCCESS.

3) If the SEK was not successfully sanitized, then the EKP_Sanitize_SEK method SHALL fail with a return status of FAIL.

4.2.1.6 EKP_Attested_Report

Start of Informative Comment

The EKP_Attested_Report method provides an attested report on the EKP feature states. A freshness nonce is provided as input and will be included as part of the report.

The EKP_Attested_Report method can be invoked at any time.

End of Informative Comment

The EKP_Attested_Report method is an Admin Template-specific method.

The following pseudo-code is the signature of the EKP_Attested_Report method (see 4.2.1.6.3 for more information).

```
EpochKeyPurgeObjectUID. EKP_Attested_Report [
    nonce : bytes,
    certuid : uidref,
    ekp_report_type = uinteger]

=>
[ ekp_result : bytes ]
```

Method UID: 00 00 00 06 00 00 0C 06

4.2.1.6.1 Parameter Descriptions

4.2.1.6.1.1 nonce

The nonce parameter specifies a 32-byte random value provided by the host to be included in the report for the method return value.

4.2.1.6.1.2 certuid

The certuid parameter specifies the certificate uidref of the certificate installed in the SD that SHALL be used for signing when the SD returns the COSE protect payload containing the EKP evidence. The uidref SHALL be associated with a certificate in the EKP_Cert_List column of the `EKPCertList` object.

4.2.1.6.1.3 ekp_report_type

The optional ekp_report_type parameter specifies the type of EKP evidence report (see Table 3) to be returned by the `EKP_Attested_Report` method. If the ekp_report_type parameter is not included in the method invocation, the SD SHALL use a CBOR-encoded byte stream (0h) for the EKP evidence report.

Table 3 Report Table Description

Value	Description
0x0000	CBOR-encode byte stream (see 5.5)
0x0001 to 0xFFFF	Reserved

4.2.1.6.2 Returned Value Descriptions

4.2.1.6.2.1 ekp_result

Start of Informative Comment

The `EKP_Attested_Report` method returns a byte string that represents a COSE protected CBOR-encoded bytes stream of the EKP Evidence (5.5) is estimated to be less than 100 bytes for an SD that supports 16 OTP slots. The COSE_Sign1 structure (5.5.2) adds a handful of bytes plus the bytes for the signature (e.g., 64 bytes for ECDSA w/ SHA-256, Curve P-256; 4627 bytes for ML-DSA-87).

End of Informative Comment

If the `EKP_Attested_Report` method returns successfully, the ekp_result returned parameter contains the signed EKP evidence report (see 5.5) as a byte string.

4.2.1.6.3 EKP_Attested_Report Method Operation

Start of Informative Comment

The nonce parameter needs to be checked to make sure 32 bytes are provided.

End of Informative Comment

If the nonce provided as a parameter does not consist of 32 bytes, then the `EKP_Attested_Report` method SHALL fail with a status of INVALID_PARAMETER.

Start of Informative Comment

The optional ekp_report_type parameter needs to be checked, if it is supplied. The current option is 0x0000 (COSE protected CBOR-encoded report).

End of Informative Comment

If ekp_report_type provided as a parameter and the value is not 0x0000, then the `EKP_Attested_Report` method SHALL fail with a status of INVALID_PARAMETER.

Start of Informative Comment

The certuid parameter needs to be checked to ensure that it refers to a usable certificate.

End of Informative Comment

If certuid is not a valid reference included in the EKP_Cert_List of the EKPCertList object, then the EKP_Attested_Report method SHALL fail with a status of INVALID_PARAMETER.

Start of Informative Comment

The EKP_Attested_Report method is not usable within a transaction.

End of Informative Comment

If EKP_Attested_Report method is invoked within a transaction, then the EKP_Attested_Report method SHALL fail with a status of TRANSACTION_FAILURE.

Start of Informative Comment

With the pre-conditions being met, proceed with the EKP_Attested_Report method to retrieve the attested EKP details. The supplied nonce is provided as part of the response.

End of Informative Comment

If:

a) the EKP_Attested_Report method is invoked;

b) 32 bytes supplied for the nonce;

c) the ekp_report_type is not specified or is equal to 0x0000; and

d) the certuid is included in the EKP_Cert_List of the EKPCertList object,

then the following changes SHALL be made:

1) The CBOR-encoded and signed byte stream (as specified in 5.5) SHALL be generated; and

2) The EKP_Attested_Report method SHALL succeed with a return status of SUCCESS and the return of the CBOR-encoded and signed byte stream in the ekp_result parameter.

4.2.2 Modified Methods

This section defines the modified methods that are required to support this feature set.

4.2.2.1 Set

Start of Informative Comment

The Epoch Key Purge (EKP) Feature Set consists of multiple methods that can be used to manage epoch key for the purpose of sanitize the contents of an SD. With the exception of the EKP_Attested_Report method, these EKP methods are only usable when the SD has transitioned into a special EKP mode. An SD that supports the EKP feature set can only transition into this EKP mode when the following are true:

- The Admin SP is in the Manufactured state;
- There are no active Storage SSCs;
- The SD has no active locking.

Once the SD has transitioned into this EKP mode, the EKP methods are available for use. In addition, the SD further constrains permitted operations to those that are permitted during user data removal methods [3]. The SD remains in

 © TCG 2026

this EKP mode until explicitly transitioning to normal operations (i.e., the SD can remain in the EKP mode indefinitely and not be usable for storage).

The Set method is used to transition into the EKP mode (see 4.2.2.1.1.1) and to exit the EKP mode (see 4.2.2.1.1.2).

When the Set method succeeds in setting the EKP_Allowed column to False, the EKP mode is disabled, use of the EKP methods is not allowed, and the SD returns to a normal mode of operation.

Success for the Set method to disable the EKP feature set is dependent on the following constraints:

a) An HEK has been programmed or a permanent HEK has been established;

b) An SEK has been programmed.

If there is an active or pending command/method at the time the Set method is invoked to transition into the EKP mode, the EKP_Termination_Policy column of the EpochKeyPurge object determines the action to be taken. A policy value of False causes the Set method to fail and the SD does not transition into the EKP mode; a policy value of True causes the SD to terminate all active or pending commands/methods that would normally prevent transitioning into the EKP mode and the SD then proceeds with transitions into the EKP mode, assuming other preconditions have been met.

End of Informative Comment

4.2.2.1.1 Set Method Operations

Start of Informative Comment

For the EKP feature set, the Set method is used to transition into the EKP mode (see 4.2.2.1.1.1) and to exit the EKP mode (see 4.2.2.1.1.2).

End of Informative Comment

4.2.2.1.1.1 Using Set to Enable the EKP Mode

Start of Informative Comment

The Set method is used to enter the EKP mode that enables the use of the EKP methods. Transitioning into the EKP mode is accomplished by setting the EKP_Allowed column of EpochKeyPurge object to a value of True.

When the Set method succeeds in setting the EKP_Allowed column to True, the EKP mode is enabled and the SD restricts operations to those that are permitted during user data removal methods [3] (e.g., read and write operations are not permitted).

Check to see if the SD is in a state that permits the EKP mode to be enabled. Enabling the EKP mode is only possible if the SD is not currently in the EKP mode (i.e., EKP_Allowed = False) and the Set method is seeking to change EKP_Allowed to True.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is currently set to a value of True and the Set method invocation is attempting to change the value of the EKP_Allowed column to True, then the Set method SHALL fail with a status of FAIL.

Start of Informative Comment

Check to see if an SSC is in use in the SD; if so, the SD cannot transition into the EKP mode.

An ownership check is used to determine whether an SSC is potentially in use. Note: Locking is not possible if the SSC SP is not owned, so a separate check is not needed.

End of Informative Comment

If:

a) the Locking SP is present in the Admin SP's SP table and the Locking SP is owned (see [4]); or

b) the Key Per I/O SP is present in the Admin SP's SP table and the Key Per I/O SP is owned see (see [5]),

then the `Set` method SHALL fail with a status of FAIL.

Start of Informative Comment

The `Set` method is used to enable the EKP mode, assuming preconditions have been met. Enabling the EKP mode has an impact on the functionality of the drive until the EKP feature is disabled.

The value of the EKP_Termination_Policy can impact EKP mode transition as follows:

- False: A pending command/method that would normally prevent transitioning into the EKP mode causes the `Set` method to fail.
- True: The SD can terminate a pending command/method that would normally prevent transitioning into the EKP mode. Such terminations do not include disabling an SSC and/or locking.

End of Informative Comment

If:

a) the EKP_Allowed column of the `EpochKeyPurge` object is currently set to a value of False and the `Set` method invocation is attempting to change the value of the EKP_Allowed column to True;

b) the Locking SP, if present in the Admin SP's SP table, is not owned (see [4]); and

c) the Key Per I/O SP, if present in the Admin SP's SP table, is not owned (see [5]),

then the following changes SHALL be made:

1) If the EKP_Termination_Policy of the `EpochKeyPurge` object is set to True, then:
    1. the SD SHALL terminate pending commands/methods preventing transition to the EKP mode;
    2. The value of the EKP_Allowed column of the `EpochKeyPurge` object SHALL be set to True (i.e., transition into the EKP mode);
    3. The SD SHALL constrain permitted operations to those that are permitted during user data removal methods [3]; and
    4. The `Set` method SHALL succeed with a return status of SUCCESS.
2) If the EKP_Termination_Policy of the `EpochKeyPurge` object is set to False and there are pending commands/methods preventing transition to the EKP mode, then the `Set` method SHALL fail with a status of FAIL.

4.2.2.1.1.2 Using Set to Disable the EKP Mode

Start of Informative Comment

The `Set` method is used to exit the EKP mode that disables the use of the EKP methods (except `EKP_Attested_Report`). Transitioning out of the EKP mode is accomplished by setting the EKP_Allowed column of `EpochKeyPurge` object to a value of False. However, this transition is only permitted under the following conditions:

- the HEK is either programmed or in a permanent state; and
- the SEK is programmed.

The above restriction ensures that the SD is only usable when the epoch keys are available, enabling future sanitization.

Check to see if the SD is in a state that permits the EKP mode to be disabled. Disabling the EKP mode is only possible if the SD is currently in the EKP mode (i.e., EKP_Allowed = True) and the Set method is seeking to change EKP_Allowed to False.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is currently set to a value of False and the Set method invocation is attempting to change the value of the EKP_Allowed column to False, then the Set method SHALL fail with a status of FAIL.

Start of Informative Comment

Transitioning out of the EKP mode can only occur when both an HEK and SEK have been programmed. If the epoch keys are not programmed the Set method fails and the SD is not available for storage purposes (e.g., reads or writes). If the HEK_State is HEK_PROGRAMMED_PENDING_RESET, the SD cannot transition out of the EKP mode until the SD reset has occurred.

End of Informative Comment

If

a) the HEK_State column of the EpochKeyPurge object is set to a value other than HEK_PROGRAMMED or HEK_PERMANENT; or

b) the SEK_State column of the EpochKeyPurge object is set to a value other than SEK_PROGRAMMED,

then the Set method SHALL fail with a status of FAIL.

Start of Informative Comment

The Set method is used to disable use of the EKP mode, assuming the preconditions that an HEK and SEK have been programmed. Disabling the EKP mode returns the drive to normal operations.

An important aspect of this transition is to ensure that all previous MEKs are no longer available (unloaded from cache and encryption engines). In addition, the SD needs to be usable, so any needed MEKs need to be generated and put into use as part of this transition.

End of Informative Comment

If:

a) the EKP_Allowed column of the EpochKeyPurge object is currently set to a value of True and the Set method invocation is attempting to change the value of the EKP_Allowed column to False;

b) the HEK_State column of the EpochKeyPurge object is set to a value of HEK_PROGRAMMED, or HEK_PERMANENT; and

c) the SEK_State column of the EpochKeyPurge object is set to a value of SEK_PROGRAMMED,

then the following changes SHALL be made:

1) The value of the EKP_Allowed column of the EpochKeyPurge object SHALL be set to False;

2) One or more MEKs, cryptographically bound to the active HEK and SEK, SHALL be generated and/or protected and used by the encryption engine of the SD (i.e., the SD is able to protect user data); and

3) The Set method SHALL succeed with a return status of SUCCESS.

 © TCG 2026

4.2.2.2 Activate

Start of Informative Comment

Use of the Activate method is not permitted when the EKP mode is active. This ensures the epoch keys are available for use by the SD.

End of Informative Comment

If the EKP_Allowed column of the EpochKeyPurge object is set to a value of True and the Activate method is invoked, then the Activate method SHALL fail with a status of FAIL.

4.2.2.3 User Data Removal Methods

Start of Informative Comment

User data removal methods [3] are expected to operate as normal, but their invocation has no impact on the EKP feature set. This means that the values of the HEK and SEK epoch keys are not changed nor are any of the values or states in the EpochKeyPurge object affected.

End of Informative Comment

With the exception of sanitization of epoch keys (i.e., this feature set) invocation of any user data removal method SHALL NOT:

a) affect the value of the HEK or SEK;

b) affect the state or values of the EpochKeyPurge object; and

c) affect the digital certificates used for signing of the EKP evidence.

4.3 Tables

This section defines new tables and modifications to existing tables required for this feature set.

4.3.1 New Tables

4.3.1.1 EpochKeyPurge (Object Table)

Start of Informative Comment

The EpochKeyPurge object is used to track the states and metadata associated with the EKP feature set.

End of Informative Comment

The EpochKeyPurge object SHALL contain information pertaining to the EKP feature set.

The contents of the EpochKeyPurge object SHALL be persistent throughout the life of the SD (i.e., not affected by storage sanitization operations).

Table 4 EpochKeyPurge Table Description

Column Number	Column Name	IsUnique	Column Type
0x00	UID	(NR)[1]	uid
0x01	Name	Yes	name
0x02	CommonName	Yes	name
0x03	EKP_Termination_Policy	(NR)[1]	boolean
0x04	EKP_Allowed	(NR)[1]	boolean
0x05	MaxSupported_HEK_Sanitizations	(NR)[1]	uinteger_2

 © TCG 2026

Column Number	Column Name	IsUnique	Column Type
0x06	Remaining_HEK_Sanitizations	(NR)[1]	uinteger_2
0x07	HEK_State	(NR)[1]	uinteger_2
0x08	HEK_Size	(NR)[1]	uinteger_2
0x09	SEK_State	(NR)[1]	uinteger_2
0x10	SEK_Size	(NR)[1]	uinteger_2
0x11	HEK_State_List	(NR)[1]	list [MaxSupported_HEK_Sanitizations]{ uinteger_2}

Notes:
1. The content of the cell is defined by the TCG Core Specification [2].

4.3.1.1.1 UID
This is the unique identifier of this row of the `EpochKeyPurge` object.

This column SHALL NOT be modifiable by the host.

4.3.1.1.2 Name
This is the name of the `EpochKeyPurge` object.

This column SHALL NOT be modifiable by the host.

4.3.1.1.3 CommonName
This is a name that MAY be shared among multiple `EpochKeyPurge` object.

This column SHALL NOT be modifiable by the host.

4.3.1.1.4 EKP_Termination_Policy
This column identifies whether or not an SD can forcibly terminate commands/methods that are actively preventing the transition into the EKP mode (i.e., invocation of the `Set` method change the value of the EKP_Allowed column of the `EpochKeyPurge` object to True). This column is modified/accessed via the methods `Set` and `Get`. The following are the expected SD behaviors upon invocation of the `Set` method for EKP_Termination_Policy column settings:

a) A value of True indicates that the policy is to terminate pending commands/methods that are preventing the transition into the EKP mode.

b) A value of False indicates that the policy is to not terminate pending commands/methods that are preventing the transition into the EKP mode.

Table 5 EKP_Termination_Policy Column Description

Value	Description
False	Do not forcibly terminate active or pending commands/methods to enter the EKP mode.
True	Forcibly terminate active or pending commands/methods to enter the EKP mode.

The default policy is False (no forcible termination). The EKP_Termination_Policy remains in effect until it is changed with the `Set` method.

4.3.1.1.5 EKP_Allowed
This column identifies whether or not EKP changes are allowed. This column is modified/accessed via the methods `Set` and `Get`. The following are the expected SD behaviors upon invocation of the `Set` method for EKP_Allowed column settings:

 © TCG 2026

a) A value of True causes the SD to immediately attempt to transition into a mode (EKP mode) where EKP methods MAY be used. If the value of the EKP_Termination_Policy column of the EpochKeyPurge object is set to True, then the SD terminates pending commands/methods that would otherwise prevent the SD from transitioning into the EKP mode. If the `Set` method is successful, the SD further constrains permitted operations to those that are permitted during user data removal methods [3]. This `Set` method is only permitted when the Admin SP is in the Manufactured state and there are no active SSCs and/or locking is not enabled within the SD.

b) A value of False causes the SD to immediately attempt to transition out of the mode where EKP methods (except `EKP_Attested_Report`) cannot be used. If the `Set` method is successful, the SD returns to its normal operating state and the EKP methods (4.2.1) SHALL NOT be used. This `Set` method is only permitted once the EKP Hard Epoch Key (HEK) and EKP Soft Epoch Key (SEK) have been successfully programmed or reprogrammed using the EKP methods (4.2.1).

Table 6 EKP_Allowed Column Description

Value	Description
False	The EKP changes are not allowed. Epoch SEK and HEK methods cannot be used.
True	The EKP changes are allowed. Epoch SEK and HEK methods can be used.

The EKP_Allowed remains in effect until it is changed with the `Set` method.

4.3.1.1.6 MaxSupported_HEK_Sanitizations

Start of Informative Comment

The MaxSupported_HEK_Sanitizations is set by the manufacturer and represents the total number of OTP slots available for programming and sanitization throughout the life of the SD. The value of this column is used by Level 0 discovery (5.2.1.5). MaxSupported_HEK_Sanitizations does not include any OTP slots that the manufacturer reserves for the manufacturing process.

End of Informative Comment

This is the maximum number of HEK sanitizations supported. This value is set by the vendor at the time of manufacturing and it does not change throughout the life of the SD.

This column SHALL NOT be modifiable by the host.

4.3.1.1.7 Remaining_HEK_Sanitizations

Start of Informative Comment

The Remaining_HEK_Sanitizations represents the number of unused OTP slots that are still available for programming and sanitization. The manufacturer sets the initial value, which is either equal to MaxSupported_HEK_Sanitizations (HEK is not programmed) or MaxSupported_HEK_Sanitizations-1 (HEK is programmed).

End of Informative Comment

This is the number of remaining HEK sanitizations that can potentially be performed on this SD. The value for this column changes with successful invocation of the `EKP_Program_Next_HEK` method. A value equal to the MaxSupported_HEK_Sanitizations is an indication that an HEK has not been programmed in the SD. A value of zero is an indication that all the HEK OTP slots have been consumed.

This column SHALL NOT be modifiable by the host.

4.3.1.1.8 HEK_State

This is the state of the EKP Hard Epoch Key (HEK) usage within the SD. The possible states are:

a) An OTP HEK has not been programmed, which is an indication that the EKP feature set is not fully active. With the exception of Level 0 discovery and the `EKP_Attested_Report` method, the EKP feature set is not in use. Any data written to the SD while the HEK has not been programmed cannot be sanitized (cryptographic erase) using the `EKP_Sanitize_OTP_HEK` method. If the SD transitions into the EKP mode when the HEK is in this state, the SD SHALL program the HEK and SEK before the SD is permitted to transition out of the EKP mode.

b) A randomized HEK has been programmed into the current OTP slot; this programming either occurred during manufacturing or by using the `EKP_Program_OTP_HEK` method (4.2.1.1). Any user data written to the SD after the HEK was programmed can be sanitized (cryptographic erase) using the `EKP_Sanitize_OTP_HEK` method. This is the normal operating mode of an SD with EKP functionality.

c) The last HEK OTP slot has been consumed (i.e., no further HEK OTP slot operations are possible) and the SD has transitioned to the use of a permanent HEK. The SD cannot be sanitized (cryptographic erase) using OTP-based sanitization (i.e., the `EKP_Sanitize_OTP_HEK` method cannot be used), but sanitization is still possible by sanitizing the SEK.

d) The current HEK OTP bank slot has been sanitized, using the `EKP_Sanitize_OTP_HEK` method (4.2.1.2). To transition out of the EKP mode, the SD SHALL program an OTP HEK or transition to the use of a permanent HEK.

e) The current HEK OTP slot is corrupted as a result of a failed invocation of the `EKP_Program_OTP_HEK` method (4.2.1.1) or the `EKP_Sanitize_OTP_HEK` method (4.2.1.2). To transition out of the EKP mode, the SD SHALL address this corruption (e.g., retrying the failed method, programming the next HEK, or transitioning to a permanent HEK).

f) An HEK has been programmed in the current OTP slot, but an SD reset is required for it to take effect. The SD SHALL be reset to address this pending situation. The SD cannot transition out of the EKP mode in this state.

g) The HEK in the current OTP slot has been sanitized, but an SD reset is required for it to take effect. The SD can be reset to fully recognize that the HEK has been sanitized; however, such a reset MAY NOT be necessary because the SD needs to either program an OTP HEK or use a permanent HEK to transition out of the EKP mode.

This column SHALL NOT be modifiable by the host.

Table 7 HEK_State Table Description

Value	State	Description
0x0000	HEK_UNUSED	An HEK has not yet been programmed. This state is an indication that the manufacturer did not program the epoch keys and that the EKP feature set cannot be used to sanitize user data.
0x0001	HEK_PROGRAMMED	An HEK is programmed in the current OTP slot.
0x0002	HEK_PROGRAMMED_PENDING_RESET	An HEK is programmed in the current OTP slot, but the HEK is not usable until the SD is reset.
0x0003	HEK_PROGRAMMED_CORRUPTED	The current OTP HEK slot is corrupted due to a failure or an issue during the attempt to program an HEK.

Value	State	Description
0x0004	HEK_PERMANENT	The OTP slots have been consumed and the SD has transitioned to the use of a permanent HEK.
0x0005	HEK_SANITIZED	The current HEK OTP slot has been sanitized.
0x0006	HEK_SANITIZED_PENDING_RESET	The current HEK OTP slot has been sanitized, but an SD reset is necessary to transition the state to HEK_SANITIZED.
0x0007	HEK_SANITIZED_CORRUPTED	The current HEK OTP slot is corrupted due to a failure or an issue during the attempt to sanitize the HEK.
0x0008 to 0xFFFF	Reserved	Reserved

4.3.1.1.9 HEK_Size

This is the size in bits of the HEK used by the SD. This value is set by the vendor at the time of manufacturing and it does not change throughout the life of the SD. This column is accessed via the `Get` method.

This column SHALL NOT be modifiable by the host.

4.3.1.1.10 SEK_State

This is the state of the EKP Soft Epoch Key (SEK) usage within the SD. The possible states are:

a) The SEK has not been programmed, which is an indication that the EKP feature set is not fully active. Any data written to the SD while the SEK has not been programmed cannot be sanitized (cryptographic erase) using the `EKP_Sanitize_SEK` method. If the SD transitions into the EKP mode when the SEK is in this state, the SD SHALL program the SEK before the SD is permitted to transition out of the EKP mode.

b) A randomized SEK has been programmed into non-volatile storage (e.g., flash); this programming either occurred during manufacturing or by using the `EKP_Program_SEK` method (4.2.1.4). User data can be sanitized (cryptographic erase) using SEK-based methods. This is the normal operating mode of an SD with EKP functionality.

c) The current SEK has been sanitized, using the `EKP_Sanitize_SEK` method (4.2.1.5).

This column SHALL NOT be modifiable by the host.

Table 8 SEK_State Table Description

Value	State	Description
0x0000	SEK_UNUSED	An SEK has not yet been programmed. This state is an indication that the manufacturer did not program the SEK and that the SEK cannot be used to sanitize user data.
0x0001	SEK_PROGRAMMED	An SEK has been programmed.
0x0002	SEK_SANITIZED	The current SEK has been sanitized.
0x0003 to 0xFFFF	Reserved	Reserved

4.3.1.1.11 SEK_Size

This is the size in bits of the SEK used by the SD. This value is set by the vendor at the time of manufacturing and it does not change throughout the life of the SD. This column is accessed via the `Get` method.

This column SHALL NOT be modifiable by the host.

4.3.1.1.12 HEK_State_List

Start of Informative Comment

HEK_State_List is an array of state values that correspond to the OTP slots used for the HEK. The manufacturer sets the state for each element to UNUSED; if the manufacturer programs the initial HEK, HEK_State_List[1] is set to PROGRAMMED.

For this document, array indices are assumed to be in the range of 1 to MaxSupported_HEK_Sanitizations.

End of Informative Comment

This list consists of MaxSupported_HEK_Sanitizations elements. Each element of the list is set to one of the state values shown in Table 9. This list provides a snapshot of the SD usage of OTP HEKs throughout the life of the drive (i.e., past, current, and yet to be used HEK entries). The possible states of the list elements are:

a) The associated HEK has not been used. If all the entries in the list are unused, this is an indication that the epoch keys (HEK and SEK) were not programmed during manufacturing.

b) The associated HEK is programmed and in use (see 4.2.1.1).

c) The attempt to program the associated HEK was not successful and resulted in a corrupted OTP HEK (see 4.2.1.1). Such a corrupted HEK cannot be used by the SD (i.e., no user data can be written), so there is no indication of a failed sanitization operation.

d) The associated HEK had been used and was subsequently sanitized (see 4.2.1.2).

e) The associated OTP HEK had been used, subsequent attempts to sanitize it were not successful, and the resulting state has been set to a corrupted sanitization (see 4.2.1.2). Such a corrupted HEK is an indication that the OTP-based sanitization has been called into question, but user data has been sanitized by the SEK sanitization, which is a prerequisite to sanitizing an HEK.

This column SHALL NOT be modifiable by the host.

Table 9 HEK_State_List Table Description

Value	State	Description
0x0000	UNUSED	The associated HEK OTP slot has not been used.
0x0001	PROGRAMMED	The associated HEK OTP slot is currently programmed (in use).
0x0002	PROGRAMMED_CORRUPTED	The associated HEK OTP slot was used but programming was not successful.
0x0003	SANITIZED	The associated HEK OTP slot was used and was properly sanitized.
0x0004	SANITIZED_CORRUPTED	The associated HEK OTP slot was used and subsequent sanitization was not successful.
0x0005 to 0xFFFF	Reserved	Reserved.

4.3.1.2 EKPCertList (Object Table)

Start of Informative Comment

The `EKPCertList` object is used to identify a list of certificates that can potentially be used for signing the EKP evidence. A host is expected to perform a `Get` on one or more of the certificates in the list, inspect the algorithms used in each certificate, select a certificate for signing purposes, validate the chosen certificate chain, and then specify the uidref of the selected certificate in a subsequent invocation of the `EKP_Attested_Report` method.

End of Informative Comment

The `EKPCertList` object SHALL contain certificate information pertaining to the EKP feature set.

Table 10 EKPCertList Table Description

Column Number	Column Name	IsUnique	Column Type
0x00	UID	(NR)[1]	uid
0x01	Name	Yes	name
0x02	CommonName	Yes	name
0x03	Max_Certificates	(NR)[1]	uinteger_2
0x04	EKP_Cert_List	(NR)[1]	list [Max_Certificates]{ ekp_certificates_object_uidref }

Notes:
1. The content of the cell is defined by the TCG Core Specification [2].

4.3.1.2.1 UID
This is the unique identifier of this row of the `EKPCertList` object.

This column SHALL NOT be modifiable by the host.

4.3.1.2.2 Name
This is the name of the `EKPCertList` object.

This column SHALL NOT be modifiable by the host.

4.3.1.2.3 CommonName
This is a name that MAY be shared among multiple `EKPCertList` object.

This column SHALL NOT be modifiable by the host.

4.3.1.2.4 Max_Certificates
This is the number of certificates in the EKP_Cert_List. The SD SHALL support at least one certificate for signing.

This column SHALL NOT be modifiable by the host.

4.3.1.2.5 EKP_Cert_List
This list consists of Max_Certificates elements. Each element of the list is set to a uidref to `Certificates` objects.

This column SHALL NOT be modifiable by the host.

4.3.2 Modified Tables
There are no tables modified by this feature set.

4.4 Types

This section defines new types and modifications to existing types required for this feature set.

4.4.1 New Types

4.4.1.1 Restricted Type

4.4.1.1.1 ekp_certificates_object_uidref

The ekp_certificates_object_uidref type is a restricted reference type that SHALL be used specifically for uidrefs to `Certificates` objects.

Table 11: ekp_certificates_object_uidref

UID	Name	Format
00 00 00 05 00 00 22 01	ekp_certificates_object_uidref	Restricted_Reference_Type{6}, uidref {CertificatesTableUID}

4.4.2 Modified Types

There are no types modified by this feature set.

 © TCG 2026

5 Feature Set Requirements

This section defines the requirements for the Epoch Key Purge (EKP) Feature Set.

5.1 Requirements Overview

The Epoch Key Purge (EKP) Feature Set consists of sanitization capabilities that MAY be implemented in a TPer. A host discovers the Epoch Key Purge specific capabilities and properties of a TPer by examining the Level 0 discovery feature descriptor.

5.2 Level 0 Discovery

A SD implementing the Feature Set SHALL:

a) Return this Epoch Key Purge feature descriptor in the Level 0 discovery response indicating the feature set is supported, enabled, and this feature set cannot be disabled.

5.2.1 Epoch Key Purge Feature Descriptor (Feature Code = 0x040D)

The contents of the feature descriptor are defined in this section below.

Table 12: Level 0 Discovery – Epoch Key Purge Feature Descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			Feature Code				
1								(LSB)
2	Feature Descriptor Version Number				Feature Set Minor Version Number			
3	Length							
4	(MSB)			Maximum Number of HEK Sanitizations Supported				
5								(LSB)
6	SD Reset Policy for OTP HEK							
7				Reserved				
11								

5.2.1.1 Feature Code

The Feature Code field value SHALL be set to `0x040D`.

5.2.1.2 Feature Descriptor Version Number

The Feature Descriptor Version Number field SHALL be set to `0x1`.

5.2.1.3 Feature Set Minor Version Number

The Feature Set Minor Version Number represents the minor version of the Epoch Key Purge (EKP) Feature Set supported by the SD.

The Feature Set Minor Version Number field SHALL be set to a value as specified in Table 13

Table 13: Feature Set Minor Versions

Feature Set Minor Version Number	Specification Referenced
0x0	Epoch Key Purge (EKP) Feature Set v1.00
All others	Reserved

5.2.1.4 Length

The Length field indicates the number of bytes in the descriptor following byte 3. The value of the Length field SHALL be 8.

5.2.1.5 Maximum Number of HEK Sanitizations Supported

The Maximum Number of HEK Sanitizations Supported field indicates the maximum number of HEKs that can be programmed over the life of the SD. This value corresponds to the MaxSupported_HEK_Sanitizations column of the `EpochKeyPurge` object.

The value of this field SHALL be set to a value greater than 0.

5.2.1.6 SD Reset Policy for OTP HEK

The SD Reset Policy for OTP HEK field indicates the policy associated with resetting the SD to ensure changes to an OTP HEK are brought into effect. If the SD Reset Policy for OTP HEK field has a value of 0x0, then the SD does not require a reset for a programmed or sanitized OTP HEK to take effect.

If the SD Reset Policy for OTP HEK field has a value other than 0x0, the SD does require a reset for a programmed or sanitized OTP HEK to take effect. In such a case, the drive MAY support multiple reset types [2] that can be used and the value of the SD Reset Policy for OTP HEK field can be inspected to determine the options, which are shown in Table 14.

Table 14: SD Reset Policy for OTP HEK Reset Types

Bit / Byte	7	6	5	4	3	2	1	0
6	Vendor Unique	Reserved	Reserved	Reserved	Programmatic	Reserved	Hardware	Power Cycle

A non-zero bit indicates that the corresponding SD reset can be used to complete the programming of an HEK.

5.3 Admin SP

This feature set modifies the behavior of the `Set` method (see 4.2.2.1).

A SD implementing this feature set SHALL support the additions to the Admin SP specified in this section, in addition to the Admin SP requirements defined in [2].

5.3.1 Tables Preconfiguration

5.3.1.1 MethodID

The MethodID Table is defined in [2]. In addition to the requirements in [4][5][6][7][8], the `MethodID` Table Preconfiguration data SHALL be modified as specified in

Table 15.

 © TCG 2026

Table 15: AdminSP - MethodID Table Preconfiguration

UID[RF]	Name[1]	CommonName	TemplateID
00 00 00 06 00 00 0C 01	“EKP_Program_OTP_HEK”	See note 2	See note 2
00 00 00 06 00 00 0C 02	“EKP_Sanitize_OTP_HEK”	See note 2	See note 2
00 00 00 06 00 00 0C 03	“EKP_Enable_Permanent_HEK”	See note 2	See note 2
00 00 00 06 00 00 0C 04	“EKP_Program_SEK”	See note 2	See note 2
00 00 00 06 00 00 0C 05	“EKP_Sanitize_SEK”	See note 2	See note 2
00 00 00 06 00 00 0C 06	“EKP_Attested_Report”	See note 2	See note 2

Notes:
1. The text in each table cell in the column is informative. See the TCG Core Specification [2].
2. The content of the cell is defined by the TCG Core Specification [2].
3. RF: The content of each cell in the column is Read-only as specified in this feature set.

5.3.1.2 EpochKeyPurge table

The `EpochKeyPurge` Table (see section 4.3.1.1) SHALL be supported and contain one row.

The `EpochKeyPurge` Table is defined in 4.3.1.1. The `EpochKeyPurge` Table Preconfiguration data is defined in Table 16.

Table 16: AdminSP - EpochKeyPurge Table Preconfiguration

UID[RF]	Name[RF]	CommonName	EKP_Termination_Policy	EKP_Allowed	MaxSupported_HEK_Sanitizations	Remaining_HEK_Sanitizations	HEK_State	HEK_Size	SEK_State	SEK_Size	HEK_State_List
00 00 1D 01 00 00 00 01	“EKP_Evidence”	See note 2	F	F	VU	VU	VU	VU	VU	VU	VU

 © TCG 2026

UID[RF]	Name[RF]	CommonName	EKP_Termination_Policy	EKP_Allowed	MaxSupported_HEK_Sanitizations	Remaining_HEK_Sanitizations	HEK_State	HEK_Size	SEK_State	SEK_Size	HEK_State_List
Notes: 1. The text in each table cell in the column is informative. See the TCG Core Specification [2]. 2. The content of the cell is defined by the TCG Core Specification [2]. 3. RF: The content of each cell in the column is Read-only as specified in this feature set.											

5.3.1.3 EKPCertList table

The `EKPCertList` Table SHALL be supported and contain one row.

The `EKPCertList` Table is defined in 4.3.1.2. The `EKPCertList` Table Preconfiguration data is specified in Table 17.

Table 17: AdminSP - EKPCertList Table Preconfiguration

UID[RF]	Name[RF]	CommonName	Max_Certificates	EKP_Cert_List
00 00 1D 02 00 00 00 01	“EKP_Cert_List1”	See note 2	VU	VU
Notes: 1. The text in each table cell in the column is informative. See the TCG Core Specification [2]. 2. The content of the cell is defined by the TCG Core Specification [2]. 3. RF: For table cells, the content of the cell is Read-only as specified by the applicable TCG SSC. For table columns, the content of each cell in the column is Read-only as specified by the applicable TCG SSC.				

5.3.1.4 Table Table

The Table Table is defined in [2]. The `Table` Table Preconfiguration data SHALL be modified from the requirement in [4][5][6][7][8] as specified in Table 18.

Table 18: AdminSP - Table Table Preconfiguration

UID[RF]	Name[RF]	CommonName	TemplateID	Kind[RF]	Column	NumColumns	Rows	RowsFree	RowBytes	LastId	MinSize	MaxSize	MandatoryWrite	RecommendedAccess
00 00 00 01 00 00 1D 01	“EpochKeyPurge”	See note 2	(NR)[3]	Object	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	0[RF]	0[RF]
00 00 00 01 00 00 1D 02	“EKPCertList”	See note 2	(NR)[3]	Object	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	0[RF]	0[RF]
00 00 00 01 00 00 00 0A	“Certificates”	See note 2	(NR)[3]	Object	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	0[RF]	0[RF]
00 00 00 01 00 00 1E 01	“EKPCertificateData1”	See note 2	(NR)[3]	Byte	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	VU[RFV]	VU[RFV]
00 00 00 01 00 00 1E 01 (+ YYYY)	“EKPCertificateDataYYYY”[1]	See note 2	(NR)[3]	Byte	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	(NR)[3]	VU[RFV]	VU[RFV]

UID[RF]	Name[RF]	CommonName	TemplateID	Kind[RF]	Column	NumColumns	Rows	RowsFree	RowBytes	LastId	MinSize	MaxSize	MandatoryWrite	RecommendedAccess
Notes: 1. EKPCertificateDataYYYY objects are added to the `Table` Table where YYYY can range from 1 to 1024. 2. The content of the cell is defined by the TCG Core Specification [2] and the applicable TCG SSC. 3. The content of the cell is not required. See the TCG Core Specification [2] and the applicable TCG SSC. 4. RF: For table cells, the content of the cell is Read-only as specified by the applicable TCG SSC. For table columns, the content of each cell in the column is Read-only as specified by the applicable TCG SSC. 5. RFV: The content of the cell is Read-only as specified by the applicable TCG SSC. The value of the cell is vendor unique.														

5.3.1.5 Access Control Element (ACE)

The ACE Table is defined in [2] and the ACE Table Preconfiguration data SHALL be modified as specified in Table 19.

Table 19: AdminSP - ACE Table Preconfiguration

Table Association -Informative Column	UID[RF]	Name[1]	CommonName	BooleanExpr	Columns[RF]
EpochKeyPurge					
	00 00 00 08 00 06 48 01	"ACE_EKP_SID"	See note 2	SID[RF]	All

Table Association -Informative Column	UID[RF]	Name[1]	CommonName	BooleanExpr	Columns[RF]
	00 00 00 08 00 06 48 02	"ACE_EKP_Attested_Report"	See note 2	Anybody[RF]	All
	00 00 00 08 00 06 48 03	"ACE_Set_EKP_Allowed_SID"	See note 2	SID[RF]	EKP_Termination_Policy, EKP_Allowed

Notes:
4. The text in each table cell in the column is informative. See the TCG Core Specification [2].
5. The content of the cell is defined by the TCG Core Specification [2].
6. RF: For table cells, the content of the cell is Read-only as specified by the applicable TCG SSC. For table columns, the content of each cell in the column is Read-only as specified by the applicable TCG SSC.

5.3.1.6 AccessControl

The AccessControl Table is defined in [2]. The `AccessControl` Table Preconfiguration data SHALL be modified as specified in Table 20:

Table 20: AdminSP - AccessControl Table Preconfiguration

Table Association - informative only	UID	InvokingID[RF]	InvokingID Name - informative only[1]	MethodID[RF]	CommonName	ACL[RF]	Log	AddACEACL	RemoveACEACL	GetACLACL[RF]	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
EpochKeyPurge																
	(ND)[3]	00 00 1D 01 00 00 00 00	EpochKeyPurge	Next	See note 2	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]
	(ND)[3]	00 00 1D 01 00 00 00 01	EKP_Evidence	Get	See note 2	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]
	(ND)[3]	00 00 1D 01 00 00 00 01	EKP_Evidence	Set	See note 2	ACE_Set_EKP_Allowed_SID	(NR)[4]	(NR)[4]	(NR)[4]	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]

 © TCG 2026

			Table Association - informative only
(ND)[3]	(ND)[3]	(ND)[3]	**UID**
00 00 1D 01 00 00 00 01	00 00 1D 01 00 00 00 01	00 00 1D 01 00 00 00 01	**InvokingID[RF]**
EKP_Evidence	EKP_Evidence	EKP_Evidence	**InvokingID Name - informative only[1]**
EKP_Enable_Permanent_HEK	EKP_Sanitize_OTP_HEK	EKP_Program_OTP_HEK	**MethodID[RF]**
See note 2	See note 2	See note 2	**CommonName**
ACE_EKP_SID	ACE_EKP_SID	ACE_EKP_SID	**ACL[RF]**
(NR)[4]	(NR)[4]	(NR)[4]	**Log**
(NR)[4]	(NR)[4]	(NR)[4]	**AddACEACL**
(NR)[4]	(NR)[4]	(NR)[4]	**RemoveACEACL**
ACE_Anybody	ACE_Anybody	ACE_Anybody	**GetACLACL[RF]**
(NR)[4]	(NR)[4]	(NR)[4]	**DeleteMethodACL**
(NR)[4]	(NR)[4]	(NR)[4]	**AddACELog**
(NR)[4]	(NR)[4]	(NR)[4]	**RemoveACELog**
(NR)[4]	(NR)[4]	(NR)[4]	**GetACLLog**
(NR)[4]	(NR)[4]	(NR)[4]	**DeleteMethodLog**
(NR)[4]	(NR)[4]	(NR)[4]	**LogTo**

© TCG 2026

			Table Association - informative only
(ND)[3]	(ND)[3]	(ND)[3]	**UID**
00 00 1D 01 00 00 00 01	00 00 1D 01 00 00 00 01	00 00 1D 01 00 00 00 01	**InvokingID[RF]**
EKP_Evidence	EKP_Evidence	EKP_Evidence	**InvokingID Name - informative only[1]**
EKP_Attested_Report	EKP_Sanitize_SEK	EKP_Program_SEK	**MethodID[RF]**
See note 2	See note 2	See note 2	**CommonName**
ACE_EKP_Attested_Report	ACE_EKP_SID	ACE_EKP_SID	**ACL[RF]**
(NR)[4]	(NR)[4]	(NR)[4]	**Log**
(NR)[4]	(NR)[4]	(NR)[4]	**AddACEACL**
(NR)[4]	(NR)[4]	(NR)[4]	**RemoveACEACL**
ACE_Anybody	ACE_Anybody	ACE_Anybody	**GetACLACL[RF]**
(NR)[4]	(NR)[4]	(NR)[4]	**DeleteMethodACL**
(NR)[4]	(NR)[4]	(NR)[4]	**AddACELog**
(NR)[4]	(NR)[4]	(NR)[4]	**RemoveACELog**
(NR)[4]	(NR)[4]	(NR)[4]	**GetACLLog**
(NR)[4]	(NR)[4]	(NR)[4]	**DeleteMethodLog**
(NR)[4]	(NR)[4]	(NR)[4]	**LogTo**

© TCG 2026

		Certificates			*EKPCertList*	**Table Association - informative only**
(ND)[3]	(ND)[3]		(ND)[3]	(ND)[3]		**UID**
00 00 00 0A 00 00 08 01	00 00 00 0A 00 00 00 00		00 00 1D 02 00 00 00 01	00 00 00 01 00 00 1D 02		**InvokingID[RF]**
EKPCertificate1	Certificates		EKP_Cert_List1	EKPCertList		**InvokingID Name - informative only[1]**
Get	Next		Get	Next		**MethodID[RF]**
See note 2	See note 2		See note 2	See note 2		**CommonName**
ACE_Anybody	ACE_Anybody		ACE_Anybody	ACE_Anybody		**ACL[RF]**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**Log**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**AddACEACL**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**RemoveACEACL**
ACE_Anybody	ACE_Anybody		ACE_Anybody	ACE_Anybody		**GetACLACL[RF]**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**DeleteMethodACL**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**AddACELog**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**RemoveACELog**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**GetACLLog**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**DeleteMethodLog**
(NR)[4]	(NR)[4]		(NR)[4]	(NR)[4]		**LogTo**

© TCG 2026

Table Association - informative only	UID	InvokingID[RF]	InvokingID Name - informative only[1]	MethodID[RF]	CommonName	ACL[RF]	Log	AddACEACL	RemoveACEACL	GetACLACL[RF]	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
	(ND)[3]	00 00 00 0A 00 00 08 00 (+MMMM)	EKPCertificateMMMM	Get	See note 2	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]
CertificateData																
	(ND)[3]	00 00 1E 01 00 00 00 00	EKPCertificateData1	Get	See note 2	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]
	(ND)[3]	00 00 1E 01 00 00 00 00 (+ YYYY * 2^{32})	EKPCertificateDataYYYY	Get	See note 2	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	ACE_Anybody	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]	(NR)[4]

Notes:

1. The text in all table cells in the column is informative.
2. The content of the cell is defined by the TCG Core Specification [2] and the applicable TCG SSC.
3. The content of the cell is not defined.
4. The content of the cell is not required. See the TCG Core Specification [2] and the applicable TCG SSC.
5. RF: The content of each cell in the column is Read-only as specified by the TCG Core Specification [2].

5.3.1.7 Certificates

The `Certificates` Table is defined in [2]. In addition to requirements in [2], the `Certificates` Table Preconfiguration data SHALL be modified as specified in Table 21.

Table 21: AdminSP - Certificates Table Preconfiguration

UID[RF]	Name[1]	Common Names	CertData[RFV]	CertSize[RFV]
00 00 00 0A 00 00 08 01	“Certificate1”	See note 4	VU	VU
00 00 00 0A 00 00 08 01 (+MMMM) (See note 5)	“CertificateMMMM”	See note 4	VU	VU

Notes:
1. The text in each table cell in the column is informative. See the TCG Core Specification [2] and the applicable TCG SSC.
2. RF: For table cells, the content of the cell is Read-only as specified by section 5.3.1.6. For table columns, the content of each cell in the column is Read-only as specified by section 5.3.1.6.
3. RFV: The content of the cell is Read-only as specified by section 5.3.1.6. The value of the cell is vendor unique.
4. The content of the cell is defined by the TCG Core Specification [2] and the applicable TCG SSC.
5. `Certificates` Table’s objects are added to the table; where MMMM can range from 1 to 1023, the maximum number of certificates supported by the TPer.

5.3.1.8 CertificateData

Start of Informative Comment

The number of public key `Certificates` objects implemented to support this feature set will vary depending on target use cases of the TPer. In a minimal implementation of this feature set, the `Certificates` table will contain exactly one object.

The cryptographic identity certificate used to attest the EKP evidence is a Vendor Unique value and can have multiple valid implementations. A self-signed certificate is invalid because it provides no receiver identity verification. A simple form of attestation would be a separate device certificate provisioned during manufacturing and attested by a public Certificate Authority. A more robust implementation would implement the Implicit Identity Based Device Attestation specification (see [20]).

End of Informative Comment

The `CertificateData` table is a byte table that contains the actual certificate data of the `Certificates` table.

The encoding of the certificate data in the byte table SHALL be an X.509v3 DER certificate as defined by [19]. The X.509 public key certificate SHALL be attested by a cryptographic identity certificate bound to the SD and SHALL NOT be self-signed (i.e., validate to the Storage Device vendor’s Root Trusted Certificate Authority). In addition, the X.509 extended attributes SHALL indicate the key usage as digitalSignature.

Its size is dependent on the X.509 DER certificate object that it contains as defined by the Storage Device manufacturer.

5.4 Additional SPs

This feature set requires no additional SPs.

5.5 EKP Report Using CBOR Encoding

Start of Informative Comment

An important aspect of this feature set is the attestability of actions taken, especially those that result in purge storage sanitization operations using valid cryptographic erase techniques [9].

A host is able to ascertain the state of the epoch keys along with other information (collectively, EKP evidence) by using the `EKP_Attested_Report` method (4.2.1.6). By providing the EKP evidence claims with appropriate integrity, authentication, and confidentiality protections, a host has assurances that the confidentiality of sensitive information has been achieved through storage sanitization.

To ensure interoperability, a SD (attester) generates "attestation evidence" that describes its attributes and current state. This evidence is formatted into an object based on the Concise Binary Object Representation (CBOR), which is a data format that serializes structured data into a compact binary form. CBOR Object Signing and Encryption (COSE) builds on this to include a cryptographic signature.

The CBOR-encoded evidence is provided to a "verifier" (e.g., a host) that

1) Verifies the cryptographic signature to confirm the token's authenticity and integrity
2) Processes the evidence against a set of predefined policies to determine the device's trustworthiness.
3) Based on the verification results, the verifier makes an informed decision about the level of access or trust to grant the device.

This section defines the claims and format of the EKP evidence returned by the `EKP_Attested_Report` method.

End of Informative Comment

5.5.1 CBOR Encoding of EKP Evidence

Start of Informative Comment

The EKP evidence is represented as a CBOR map (major type 5), which is comprised of pairs of data items, each pair consists of a key that is immediately followed by a value.

In CBOR, the initial byte of each encoded data item contains both information about the major type (the high-order 3 bits and additional information (the low-order 5 bits) referred to as the argument value. For a CBOR map, the argument is the number of pairs of data items in the map. For example, a map that contains 9 pairs would have an initial byte of 101_01001b (major type 5, additional information 9 for the number of pairs) followed by the 18 remaining items. The first item is the first key, the second item is the first value, the third item is the second key, and so on.

To ensure consistent CBOR serialization and maximize interoperability across different implementations, the data pairs representing the EKP evidence, have to use the same key values for data items. The order of the pairs is not important as long as the correct key values are used (e.g., key value 9 with its data could be presented before key value 2).

To show an example of CBOR encoding of a hypothetical SD, assume the following:

- the SD has five OTP slots;
- two OTP slots have already been consumed;
- the EKP mode is active;
- the HEK has been programmed into the third slot; and
- the SEK is still in a sanitized state (i.e., not programed).

Using the CBOR diagnostic notation [10], the following presents what this EKP evidence looks like for this hypothetical SD:

```
/ ekp-map /
{
   / map label / 0: "TCG EKP Feature Set Evidence"
   / version / 1: "1.00"                          / EKP FS Version Number /
   / nonce / 2: h'372f6b03100c845a501f36ad1994b63459e59e24d50ea56179a61ae72bc22ae9'
   / ekp allowed / 3: true                        / EKP mode active /
   / max hek sanitizations / 4: h'0005'           / 5 OTP slots /
   / remaining hek sanitizations / 5: h'0002'     / 2 remaining OTP slots /
   / hek state / 6: h'0001'                       / HEK_PROGRAMMED /
   / sek state / 7: h'0002'                       / SEK_SANITIZED /
   /  hek state list / 8: [
      h'0003',                                    / OTP 1: SANITIZED /
      h'0004',                                    / OTP 2: SANITIZED_CORRUPTED /
      h'0001'                                     / OTP 3: PROGRAMMED /
      h'0000'                                     / OTP 4: UNUSED /
      h'0000'                                     / OTP 5: UNUSED /
   ]
}
```

A nicely formatted version of the binary representation of this hypothetical SD is as follows:

```
A9                          # map(9)
  00                        # unsigned(0) – Key Value 0
  78 1C                       # text(28)
    54434720454B50204665617475726520536574204576696465 6E6365 # "TCG EKP Feature Set Evidence"
  01                        # unsigned(1) – Key Value 1
  64                        # text(4)
    312E3030                    # "1.00"
  02                        # unsigned(2) – Key Value 2
  58 20                       # bytes(32)
    372F6B03100C845A501F36AD1994B63459E59E24D50EA56179A61AE72BC22AE9                    #
"7/k\u0003\u0010\f\x84ZP\u001F6\xAD\u0019\x94\xB64Y\xE5\x9E$\xD5\u000E\xA5ay\xA6\u001A\xE7+\xC2*\xE9"
  03                        # unsigned(3) – Key Value 3
  F5                        # primitive(21) – True
  04                        # unsigned(4) – Key Value 4
```

```
42                  # bytes(2)
  0005               # "\u0000\u0005"
05                  # unsigned(5) – Key Value 5
42                  # bytes(2)
  0002               # "\u0000\u0002"
06                  # unsigned(6) – Key Value 6
42                  # bytes(2)
  0001               # "\u0000\u0001"
07                  # unsigned(7) – Key Value 7
42                  # bytes(2)
  0002               # "\u0000\u0002"
08                  # unsigned(8) – Key Value 8
85                  # array(5)
  42                # bytes(2)
    0003             # "\u0000\u0003"
  42                # bytes(2)
    0004             # "\u0000\u0004"
  42                # bytes(2)
    0001             # "\u0000\u0001"
  42                # bytes(2)
    0000             # "\u0000\u0000"
  42                # bytes(2)
    0000             # "\u0000\u0000"
```

Note that CBOR uses big endian.

End of Informative Comment

If ekp_report_type is equal to 0x0000, then EKP evidence SHALL be encoded using the Concise Binary Object Representation (CBOR) [10].

If ekp_report_type is equal to 0x0000, then EKP evidence SHALL be presented in the following representation (shown as a CDDL [15] fragment):

```
EKP_Evidence = {                    ; CBOR map for the EKP evidence
    0: “TCG EKP Feature Set Evidence”,  ; 0: Map label for EKP evidence
    1:”1.00”,                       ; 1: EKP FS Version Number as string
    (2 => bstr .size 32),           ; 2: Nonce – 32-bytes
    (3 => bool),                    ; 3: EKP_Allowed (true / false)
```

```
    (4 => uint .size 2),                ; 4: MaxSupported_HEK_Sanitizations (2 bytes)
    (5 => uint .size 2),                ; 5: Remaining_HEK_Sanitizations (2 bytes)
    (6 => uint .size 2),                ; 6: HEK_State (2 bytes)
    (7 => uint .size 2),                ; 7: SEK_State (2 bytes)
    (8 => [+ uint .size 2])             ; 8: HEK State List - Array of 1 or more two-byte elements
}
```

5.5.2 COSE Protection of EKP Evidence

Start of Informative Comment

The basic COSE object structure, described in IETF RFC 9052 [11], consists of the following elements:

- COSE headers, which are encoded and wrapped in a byte stream (bstr)
  - Protected: parameters that are cryptographically protected. This feature set has no parameters that need to be protected; this can be encoded as h'a0' (empty).
  - Unprotected: parameters that are not cryptographically protected. This feature set has no parameters that are unprotected; this can be encoded as h'a0' (empty).
- Message payload, which can either be included in the message or remain detached. For this feature set, the payload is always attached. This field contains the serialized content to be signed and it is wrapped in a bstr to ensure that it is transported without changes.

This feature set uses the COSE_Sign structure for one signature to be placed on a message. The CBOR object that carries the body, the signature, and the information about the body and signature is called the COSE_Sign1 structure.

For this feature set, the COSE_Sign1 signature structure is tagged and identified by the CBOR tag 18.

The COSE_Sign1 structure is a CBOR array. The fields of the array in order are:

protected: empty for this feature set

unprotected: header map (empty for this feature set)

payload: serialized content (bstr) to be signed

signature: This field contains the computed signature value. The type of the field is a bstr.

The CDDL [15] fragment that represents the above text for COSE_Sign1 follows.

```
COSE_Sign1_Tagged = #6.18(COSE_Sign1)
COSE_Sign1 = [
    Headers,
    payload : bstr,
    signature : bstr
]
```

The payload consists of the CBOR-encoded content associated with EKP_Evidence described in 5.5.1.

End of Informative Comment

The EKP_Evidence (see 5.5.1) SHALL be protected with a cryptographic signature using CBOR Object Signing and Encryption (COSE), using the COSE_Sign1 signature structure [11].

The EKP evidence SHALL be the message payload and it SHALL NOT be encrypted.

While it is possible to specify algorithms as part of the COSE headers, the options at the time of this writing are limited, so they are not used with this feature set. Instead, hosts are expected to retrieve available SD certificates, select a certificate to be used for signing, and specify the certificate (and indirectly the algorithms) to be used for signing the payload.

The SD SHALL NOT include COSE header parameters associated with algorithms (e.g., alg, kid, IV, and partial IV).

The COSE_Sign1 structure has an option to include the certificate chain of the certificate used for signing. For this feature set, the host is expected to use the `Get` method to retrieve SD certificates, so there is no need to transmit the certificate as part of the COSE_Sign1 structure.

The SD SHALL NOT include certificates (e.g., using the x5t or x5chain label [16]) in COSE protected object delivering the EKP evidence.

 © TCG 2026