
**Information technology — Trusted
Platform Module —**

**Part 3:
Structures**

*Technologies de l'information — Module de plate-forme de confiance —
Partie 3: Structures*



Reference number
ISO/IEC 11889-3:2009(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2009

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2009

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Table of Contents

1. Scope	1
1.1 Key words	1
1.2 Statement Type	1
2. Normative references	2
3. Abbreviated Terms	3
4. Structures and Formats	5
4.1 Representation of Information	5
4.1.1 Endness of Structures	5
4.1.2 Byte Packing	5
4.1.3 Lengths	5
4.1.4 Structure Definitions	5
4.2 Defines	6
4.2.1 Basic data types	6
4.2.2 Boolean types	6
4.2.3 Helper redefinitions	6
4.2.4 Vendor specific	8
5. Structure Tags	9
5.1 TPM_STRUCTURE_TAG	10
6. Types	12
6.1 TPM_RESOURCE_TYPE	12
6.2 TPM_PAYLOAD_TYPE	13
6.3 TPM_ENTITY_TYPE	14
6.4 Handles	15
6.4.1 Reserved Key Handles	16
6.5 TPM_STARTUP_TYPE	17
6.6 TPM_STARTUP_EFFECTS	18
6.7 TPM_PROTOCOL_ID	19
6.8 TPM_ALGORITHM_ID	20
6.9 TPM_PHYSICAL_PRESENCE	21
6.10 TPM_MIGRATE_SCHEME	22
6.11 TPM_EK_TYPE	23
6.12 TPM_PLATFORM_SPECIFIC	24
7. Basic Structures	25
7.1 TPM_STRUCT_VER	25
7.2 TPM_VERSION_BYTE	26
7.3 TPM_VERSION	27
7.4 TPM_DIGEST	28

7.4.1	Creating a PCR composite hash	29
7.5	TPM_NONCE	30
7.5.1	TPM_PROOF	31
7.6	TPM_AUTHDATA	32
7.7	TPM_KEY_HANDLE_LIST	33
7.8	TPM_KEY_USAGE values	34
7.8.1	Mandatory Key Usage Schemes	34
7.9	TPM_AUTH_DATA_USAGE values	36
7.10	TPM_KEY_FLAGS	37
7.11	TPM_CHANGEAUTH_VALIDATE	38
7.12	TPM_MIGRATIONKEYAUTH	39
7.13	TPM_COUNTER_VALUE	40
7.14	TPM_SIGN_INFO Structure	41
7.15	TPM_MSA_COMPOSITE	42
7.16	TPM_CMK_AUTH	43
7.17	TPM_CMK_DELEGATE values	44
7.18	TPM_SELECT_SIZE	45
7.19	TPM_CMK_MIGAUTH	46
7.20	TPM_CMK_SIGTICKET	47
7.21	TPM_CMK_MA_APPROVAL	48
8.	TPM_TAG (Command and Response Tags)	49
9.	Internal Data Held By TPM	50
9.1	TPM_PERMANENT_FLAGS	51
9.1.1	Flag Restrictions	55
9.2	TPM_STCLEAR_FLAGS	56
9.2.1	Flag Restrictions	58
9.3	TPM_STANY_FLAGS	59
9.3.1	Flag Restrictions	60
9.4	TPM_PERMANENT_DATA	61
9.4.1	Flag Restrictions	64
9.5	TPM_STCLEAR_DATA	65
	Flag Restrictions	66
	Deferred Physical Presence Bit Map	66
9.6	TPM_STANY_DATA	67
9.6.1	Flag Restrictions	68
10.	PCR Structures	69
10.1	TPM_PCR_SELECTION	70
10.2	TPM_PCR_COMPOSITE	72
10.3	TPM_PCR_INFO	73
10.4	TPM_PCR_INFO_LONG	74

10.5	TPM_PCR_INFO_SHORT	75
10.6	TPM_LOCALITY_SELECTION	76
10.7	PCR Attributes	77
10.8	TPM_PCR_ATTRIBUTES	78
10.8.1	Comparing command locality to PCR flags	79
10.9	Debug PCR register	80
10.10	Mapping PCR Structures	81
11.	Storage Structures	83
11.1	TPM_STORED_DATA	83
11.2	TPM_STORED_DATA12	84
11.3	TPM_SEALED_DATA	85
11.4	TPM_SYMMETRIC_KEY	86
11.5	TPM_BOUND_DATA	87
12.	TPM_KEY complex	88
12.1	TPM_KEY_PARMS	89
12.1.1	TPM_RSA_KEY_PARMS	90
12.1.2	TPM_SYMMETRIC_KEY_PARMS	90
12.2	TPM_KEY	91
12.3	TPM_KEY12	92
12.4	TPM_STORE_PUBKEY	93
12.5	TPM_PUBKEY	94
12.6	TPM_STORE_ASYMKEY	95
12.7	TPM_STORE_PRIVKEY	96
12.8	TPM_MIGRATE_ASYMKEY	97
12.9	TPM_KEY_CONTROL	98
13.	Signed Structures	99
13.1	TPM_CERTIFY_INFO Structure	99
13.2	TPM_CERTIFY_INFO2 Structure	100
13.3	TPM_QUOTE_INFO Structure	101
13.4	TPM_QUOTE_INFO2 Structure	102
14.	Identity Structures	103
14.1	TPM_EK_BLOB	103
14.2	TPM_EK_BLOB_ACTIVATE	104
14.3	TPM_EK_BLOB_AUTH	105
14.4	TPM_CHOSENID_HASH	106
14.5	TPM_IDENTITY_CONTENTS	107
14.6	TPM_IDENTITY_REQ	108
14.7	TPM_IDENTITY_PROOF	109
14.8	TPM_ASYM_CA_CONTENTS	110
14.9	TPM_SYM_CA_ATTESTATION	111

15. Transport structures	112
15.1 TPM_TRANSPORT_PUBLIC	112
15.1.1 TPM_TRANSPORT_ATTRIBUTES Definitions	112
15.2 TPM_TRANSPORT_INTERNAL	113
15.3 TPM_TRANSPORT_LOG_IN structure	114
15.4 TPM_TRANSPORT_LOG_OUT structure	115
15.5 TPM_TRANSPORT_AUTH structure	116
16. Audit Structures	117
16.1 TPM_AUDIT_EVENT_IN structure	117
16.2 TPM_AUDIT_EVENT_OUT structure	118
17. Tick Structures	119
17.1 TPM_CURRENT_TICKS	119
18. Return codes	120
19. Ordinals	125
19.1 TSC Ordinals	133
20. Context structures	134
20.1 TPM_CONTEXT_BLOB	134
20.2 TPM_CONTEXT_SENSITIVE	136
21. NV storage structures	137
21.1 TPM_NV_INDEX	137
21.1.1 Required TPM_NV_INDEX values	138
21.1.2 Reserved Index values	139
21.2 TPM_NV_ATTRIBUTES	140
21.3 TPM_NV_DATA_PUBLIC	142
21.4 TPM_NV_DATA_SENSITIVE	143
21.5 Max NV Size	144
21.6 TPM_NV_DATA_AREA	145
22. Delegate Structures	146
22.1 Structures and encryption	146
22.2 Delegate Definitions	147
22.2.1 Owner Permission Settings	148
22.2.2 Owner commands not delegated	149
22.2.3 Key Permission settings	150
22.2.4 Key commands not delegated	151
22.3 TPM_FAMILY_FLAGS	152
22.4 TPM_FAMILY_LABEL	153
22.5 TPM_FAMILY_TABLE_ENTRY	154
22.6 TPM_FAMILY_TABLE	155
22.7 TPM_DELEGATE_LABEL	156
22.8 TPM_DELEGATE_PUBLIC	157

22.9	TPM_DELEGATE_TABLE_ROW	158
22.10	TPM_DELEGATE_TABLE	159
22.11	TPM_DELEGATE_SENSITIVE	160
22.12	TPM_DELEGATE_OWNER_BLOB	161
22.13	TPM_DELEGATE_KEY_BLOB	162
22.14	TPM_FAMILY_OPERATION Values	163
23.	Capability areas	164
23.1	TPM_CAPABILITY_AREA for TPM_GetCapability	164
23.2	CAP_PROPERTY SubCap values for TPM_GetCapability	167
23.3	Bit ordering for structures	169
23.3.1	Deprecated GetCapability Responses	169
23.4	TPM_CAPABILITY_AREA Values for TPM_SetCapability	170
23.5	SubCap Values for TPM_SetCapability	171
23.6	TPM_CAP_VERSION_INFO	172
23.7	TPM_DA_INFO	173
23.8	TPM_DA_INFO_LIMITED	174
23.9	TPM_DA_STATE	175
23.10	TPM_DA_ACTION_TYPE	176
24.	DAA Structures	177
24.1	Size definitions	177
24.2	Constant definitions	177
24.3	TPM_DAA_ISSUER	178
24.4	TPM_DAA_TPM	179
24.5	TPM_DAA_CONTEXT	180
24.6	TPM_DAA_JOINDATA	181
24.7	TPM_STANY_DATA Additions	182
24.8	TPM_DAA_BLOB	183
24.9	TPM_DAA_SENSITIVE	184
25.	Redirection	185
25.1	TPM_REDIR_COMMAND	185
26.	Deprecated Structures	186
26.1	Persistent Flags	186
26.2	Volatile Flags	186
26.3	TPM persistent data	186
26.4	TPM volatile data	186
26.5	TPM SV data	187
26.6	TPM_SYM_MODE	187
27.	Bibliography	188

List of Tables

Table 1: Basic data type parameters	6
Table 2: Boolean types	6
Table 3: Helper redefinition parameters	6
Table 4: Vendor specific parameters	8
Table 5: TPM_StructureTag	10
Table 6: TPM_ResourceTypes	12
Table 7: TPM_PAYLOAD_TYPE values	13
Table 8: TPM_ENTITY_TYPE LSB Values	14
Table 9: TPM_ENTITY_TYPE MSB Values	14
Table 10: Key Handle Values	16
Table 11: TPM_STARTUP_TYPE Values	17
Table 12: Types of Startup	18
Table 13: TPM_PROTOCOL_ID Values	19
Table 14: TPM_ALGORITHM_ID values	20
Table 15: TPM_PHYSICAL_PRESENCE parameters	21
Table 16: TPM_MIGRATE_SCHEME values	22
Table 17: TPM_EK_TYPE parameters	23
Table 18: TPM_PLATFORM_SPECIFIC parameters	24
Table 19: TPM_STRUCT_VER parameters	25
Table 20: TPM_VERSION_BYTE rule	26
Table 21: TPM_VERSION parameters	27
Table 22: TPM_DIGEST parameters	28
Table 23: TPM_DIGEST redefinitions	28
Table 24: TPM_NONCE parameters	30
Table 25: TPM_NONCE redefinitons	30
Table 24: TPM_PROOF parameters	31
Table 26: TPM_AUTHDATA redefinitions	32
Table 27: TPM_KEY_HANDLE_LIST parameters	33
Table 28: TPM_KEY_USAGE values	34
Table 29: Mandatory Key Usage Schemes	35
Table 30: Valid encryption schemes	35
Table 31: Valid signature schemes	35
Table 32: Combinations of TPM_AUTH_DATA_USAGE values	36
Table 33: TPM_AUTH_DATA_USAGE values	36
Table 34: TPM_KEY_FLAGS Values	37
Table 35: TPM_CHANGEAUTH_VALIDATE parameters	38
Table 36: TPM_MIGRATIONKEYAUTH parameters	39
Table 37: TPM_COUNTER_VALUE parameters	40

Table 38: TPM_SIGN_INFO Structure parameters	41
Table 39: TPM_MSA_COMPOSITE parameters	42
Table 40: TPM_CMK_AUTH parameters	43
Table 41: TPM_CMK_DELEGATE values	44
Table 42: TPM_SELECT_SIZE parameters	45
Table 43: TPM_CMK_MIGAUTH parameters	46
Table 44: TPM_CMK_SIGTICKET parameters	47
Table 45: TPM_CMK_MA_APPROVAL parameters	48
Table 46: TPM_TAG (Command and Response Tags)	49
Table 47: TPM_PERMANENT_FLAGS parameters	51
Table 48: TPM_PERMANENT_FLAGS restrictions	55
Table 49: TPM_STCLEAR_FLAGS parameters	56
Table 50: TPM_STCLEAR_FLAGS restrictions	58
Table 51: TPM_STANY_FLAGS parameters	59
Table 52: TPM_STANY_FLAGS restrictions	60
Table 53: TPM_PERMANENT_DATA parameters	62
Table 54: Flag Restrictions	64
Table 55: TPM_STCLEAR_DATA parameters	65
Table 56: Flag Restrictions	66
Table 57: Deferred Physical Presence Bit Map	66
Table 58: TPM_STANY_DATA parameters	67
Table 59: Flag Restrictions	68
Table 60: TPM_PCR_SELECTION parameters	71
Table 61: TPM_PCR_COMPOSITE parameters	72
Table 62: TPM_PCR_INFO parameters	73
Table 63: TPM_PCR_INFO_LONG parameters	74
Table 64: TPM_PCR_INFO_SHORT parameters	75
Table 65: TPM_LOCALITY_SELECTION definitions	76
Table 66: TPM_PCR_ATTRIBUTES - types of persistent data	78
Table 67: TPM_STORED_DATA parameters	83
Table 68: TPM_STORED_DATA12 parameters	84
Table 69: TPM_SEALED_DATA parameters	85
Table 70: TPM_SYMMETRIC_KEY parameters	86
Table 71: TPM_BOUND_DATA parameters	87
Table 72: TPM_KEY_PARMS parameters	89
Table 73: TPM_KEY_PARMS descriptions	89
Table 74: TPM_RSA_KEY_PARMS parameters	90
Table 75: TPM_SYMMETRIC_KEY_PARMS parameters	90
Table 76: TPM_KEY parameters	91
Table 77: TPM_KEY12 parameters	92

Table 78: TPM_STORE_PUBKEY parameters	93
Table 79: TPM_STORE_PUBKEY algorithm	93
Table 80: TPM_PUBKEY parameters	94
Table 81: TPM_STORE_ASYMKEY parameters	95
Table 82: TPM_STORE_PRIVKEY parameters	96
Table 83: TPM_STORE_PRIVKEY algorithm	96
Table 84: TPM_MIGRATE_ASYMKEY parameters	97
Table 85: TPM_KEY_CONTROL parameters	98
Table 86: TPM_CERTIFY_INFO Structure parameters	99
Table 87: TPM_CERTIFY_INFO2 Structure parameters	100
Table 88: TPM_QUOTE_INFO Structure parameters	101
Table 89: TPM_QUOTE_INFO2 Structure parameters	102
Table 90: TPM_EK_BLOB parameters	103
Table 91: TPM_EK_BLOB_ACTIVATE parameters	104
Table 92: TPM_EK_BLOB_AUTH parameters	105
Table 93: TPM_CHOSENID_HASH parameters	106
Table 94: TPM_IDENTITY_CONTENTS parameters	107
Table 95: TPM_IDENTITY_REQ parameters	108
Table 96: TPM_IDENTITY_PROOF parameters	109
Table 97: TPM_ASYM_CA_CONTENTS parameters	110
Table 98: TPM_SYM_CA_ATTESTATION parameters	111
Table 99: TPM_TRANSPORT_PUBLIC parameters	112
Table 100: TPM_TRANSPORT_ATTRIBUTES Definitions	112
Table 101: TPM_TRANSPORT_INTERNAL parameters	113
Table 102: TPM_TRANSPORT_LOG_IN structure parameters	114
Table 103: TPM_TRANSPORT_LOG_OUT structure parameters	115
Table 104: TPM_TRANSPORT_AUTH structure parameters	116
Table 105: TPM_AUDIT_EVENT_IN structure parameters	117
Table 106: TPM_AUDIT_EVENT_OUT structure parameters	118
Table 107: TPM_CURRENT_TICKS parameters	119
Table 108: Mask Parameters	121
Table 109: TPM-defined fatal error codes	122
Table 110: TPM-defined non-fatal errors	124
Table 111: Ordinal masks	125
Table 112: Ordinal purviews	126
Table 113: Ordinal combinations	126
Table 114: Column descriptions	126
Table 115: Ordinal table	127
Table 116: TSC Ordinals	133
Table 117: TPM_CONTEXT_BLOB parameters	135

Table 118: TPM_CONTEXT_SENSITIVE parameters	136
Table 119: Required TPM_NV_INDEX values	138
Table 120: Reserved Index values	139
Table 121: TPM_NV_ATTRIBUTES parameters	140
Table 122: TPM_NV_ATTRIBUTES attribute values	141
Table 123: TPM_NV_DATA_PUBLIC parameters	142
Table 124: TPM_NV_DATA_SENSITIVE parameters	143
Table 125: Delegate Definitions parameters	147
Table 126: Owner Permission Settings - Per1 bits	148
Table 127: Owner Permission Settings - Per2 bits	149
Table 128: Owner commands not delegated	149
Table 129: Key Permission settings - Per1 bits	150
Table 130: Key Permission settings - Per2 bits	151
Table 131: Key commands not delegated	151
Table 132: TPM_FAMILY_FLAGS bit settings	152
Table 133: TPM_FAMILY_LABEL parameters	153
Table 134: TPM_FAMILY_TABLE_ENTRY parameters	154
Table 135: TPM_FAMILY_TABLE parameters	155
Table 136: TPM_DELEGATE_LABEL parameters	156
Table 137: TPM_DELEGATE_PUBLIC parameters	157
Table 138: TPM_DELEGATE_TABLE_ROW	158
Table 139: TPM_DELEGATE_TABLE parameters	159
Table 140: TPM_DELEGATE_SENSITIVE parameters	160
Table 141: TPM_DELEGATE_OWNER_BLOB parameters	161
Table 142: TPM_DELEGATE_KEY_BLOB parameters	162
Table 143: TPM_FAMILY_OPERATION Values	163
Table 144: TPM_CAPABILITY_AREA Values for TPM_GetCapability	165
Table 145: TPM_CAP_PROPERTY SubCap Values for TPM_GetCapability	167
Table 146: Deprecated GetCapability Responses	169
Table 147: TPM_CAPABILITY_AREA Values for TPM_SetCapability	170
Table 148: TPM_CAP_VERSION_INFO parameters	172
Table 149: TPM_CAP_VERSION_INFO example output	172
Table 150: TPM_DA_INFO parameters	173
Table 151: TPM_DA_INFO_LIMITED parameters	174
Table 152: TPM_DA_STATE Values	175
Table 153: TPM_DA_ACTION_TYPE parameters	176
Table 154: TPM_DA_ACTION_TYPE action values	176
Table 155: TPM_DAA_ISSUER parameters	178
Table 156: TPM_DAA_TPM parameters	179
Table 157: TPM_DAA_CONTEXT parameters	180

Table 158: TPM_DAA_JOINDATA parameters	181
Table 159: TPM_STANY_DATA Additions: types of volatile data	182
Table 160: TPM_DAA_BLOB parameters	183
Table 161: TPM_DAA_SENSITIVE parameters	184
Table 162: TPM_REDIR_COMMAND	185
Table 163. TPM_SYM_MODE values	187

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 11889-3 was prepared by the Trusted Computing Group (TCG) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

ISO/IEC 11889 consists of the following parts, under the general title *Information technology — Trusted Platform Module*:

- *Part 1: Overview*
- *Part 2: Design principles*
- *Part 3: Structures*
- *Part 4: Commands*

Introduction

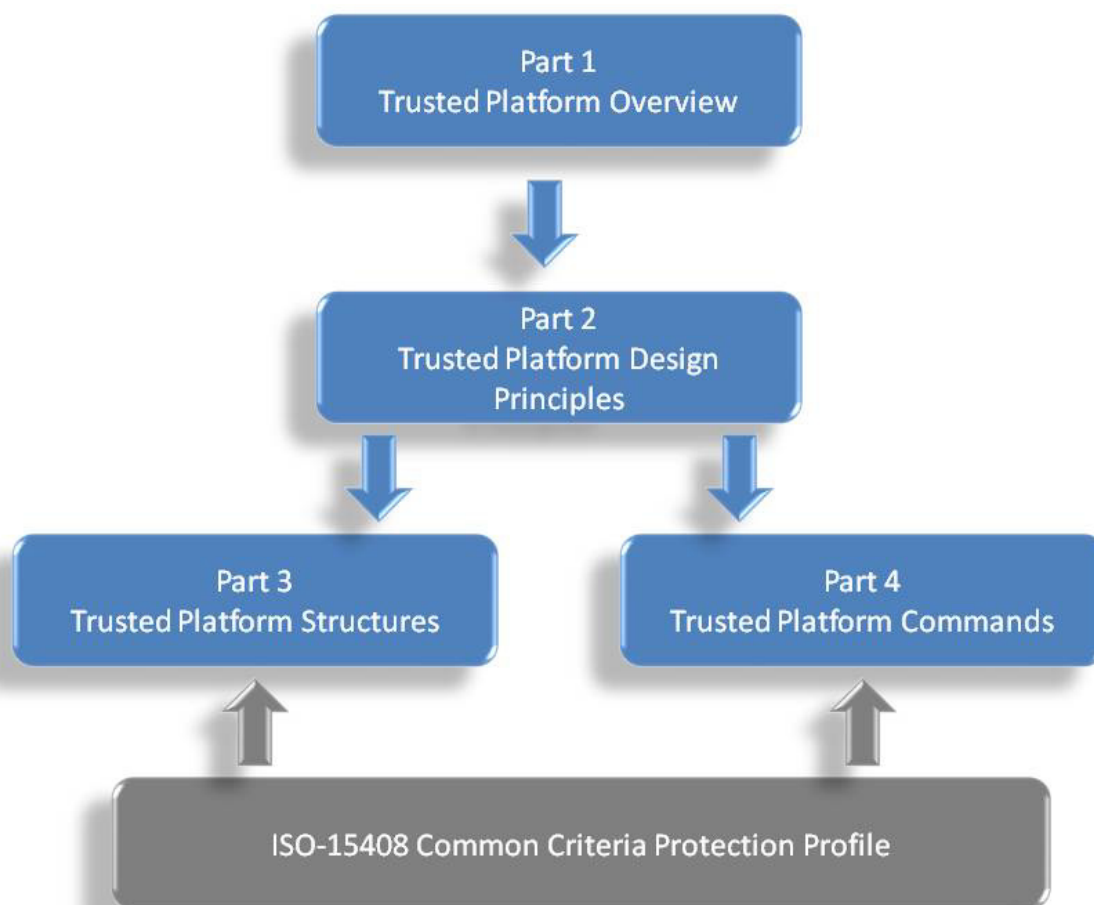


Figure 1. TPM Main Specification Roadmap

Start of informative comment

ISO/IEC 11889 is from the Trusted Computing Group (TCG) Trusted Platform Module (TPM) specification 1.2 version 103. The part numbers for ISO/IEC 11889 and the TCG specification do not match. The reason is the inclusion of the Overview document that is not a member of the TCG part numbering. The mapping between the two is as follows:

ISO Reference	TCG Reference
Part 1 Overview	Not published
Part 2 Design Principles	Part 1 Design Principles
Part 3 Structures	Part 2 Structures
Part 4 Commands	Part 3 Commands

End of informative comment

Information technology — Trusted Platform Module —

Part 3: Structures

1. Scope

ISO/IEC 11889 defines the Trusted Platform Module (TPM), a device that enables trust in computing platforms in general. ISO/IEC 11889 is broken into parts to make the role of each document clear. Any version of the standard requires all parts to be a complete standard.

A TPM designer **MUST** be aware that for a complete definition of all requirements necessary to build a TPM, the designer **MUST** use the appropriate platform specific specification to understand all of the TPM requirements.

Part 3 defines the structures and constants in use by the TPM. As the TPM must interoperate between various implementations, these structures enable the required interoperability. The other rationale for defining the structures is that some of the structures require security properties, either confidentiality or integrity calculations. If the structures are built incorrectly the security properties may not be present, hence the need to define the structures.

1.1 Key words

The key words “**MUST**,” “**MUST NOT**,” “**REQUIRED**,” “**SHALL**,” “**SHALL NOT**,” “**SHOULD**,” “**SHOULD NOT**,” “**RECOMMENDED**,” “**MAY**,” and “**OPTIONAL**” in this document’s 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. You will encounter 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, you can consider it of the kind normative statements.

For example:

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 standard the user **MUST** read the standard. (This use of **MUST** does not require any action).

End of informative comment

This is the first paragraph of one or more paragraphs (and/or sections) containing the text of the kind normative statements ...

To understand the standard the user **MUST** read the standard. (This use of **MUST** indicates a keyword usage and requires an action).

2. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 8825-1|ITU-T X.690: Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

ISO/IEC 10118-3, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions, Clause 9, SHA-1

ISO/IEC 18033-3, Information technology — Security techniques — Encryption algorithms — Part 3, Block ciphers, Clause 5.1 AES

IEEE P1363, Institute of Electrical and Electronics Engineers: Standard Specifications For Public-Key Cryptography

IETF RFC 2104, Internet Engineering Task Force Request for Comments 2104: HMAC: Keyed-Hashing for Message Authentication

IETF RFC 2119, Internet Engineering Task Force Request for Comments 2119: Key words for use in RFCs to Indicate Requirement Levels

PKCS #1 Version 2.1, RSA Cryptography Standard. This document is superseded by P1363, except for section 7.2 that defines the V1.5 RSA signature scheme in use by the TPM.

3. Abbreviated Terms

Abbreviation	Description
AACP	Asymmetric Authorization Change Protocol
ADCP	Authorization Data Change Protocol
ADIP	Authorization Data Insertion Protocol
AIK	Attestation Identity Key
AMC	Audit Monotonic Counter
APIP	Time-Phased Implementation Plan
AuthData	Authentication Data or Authorization Data, depending on the context
BCD	Binary Coded Decimal
BIOS	Basic Input/Output System
CA	Certification of Authority
CDI	Controlled Data Item
CMK	Cerifiable/Certified Migratable Keys
CRT	Chinese Remainder Theorem
CRTM	Core Root of Trust Measurement
CTR	Counter-mode encryption
DAA	Direct Autonomous Attestation
DIR	Data Integrity Register
DOS	Disk Operating System
DSA	Digital Signature Algorithm
DSAP	Delegate-Specific Authorization Protocol
ECB	Electronic Codebook Mode
EK	Endorsement Key
ET	ExecuteTransport or Entity Type
FIPS	Federal Information Processing Standard
GPIO	General Purpose I/O
HMAC	Hash Message Authentication Code
HW	Hardware Interface
IB	Internal Base
I/O	Input/Output
IV	Initialization Vector
KH	Key Handle
LEAP	Lightweight Extensible Authentication Protocol for wireless computer networks
LK	Loaded Key
LOM	Limited Operation Mode
LPC	Low Pin Count
LSB	Least Significant Byte
MA	Migration Authority/Authorization
MIDL	Microsoft Interface Definition Language
MSA	Migration Selection Authority
MSB	Most Significant Byte
NV	Non-volatile

Abbreviation	Description
NVRAM	Non-Volatile Random Access Memory
OAEP	Optimal Asymmetric Encryption Padding
OEM	Original Equipment Manufacturer
OIAP	Object-Independent Authorization Protocol
OID	Object Identifier
OSAP	Object-Specific Authorization Protocol
PCR	Platform Configuration Register
PI	Personal Information
PII	Personally Identifiable Information
POST	Power On Self Test
PRIVEK	Private Endorsement Key
PRNG	Pseudo Random Number Generator
PSS	Probabilistic Signature Scheme
PUBEK	Public Endorsement Key
RNG	Random Number Generator
RSA	Algorithm for public-key cryptography. The letters R, S, and A represent the initials of the first public describers of the algorithm.
RTM	Release to Manufacturing/Ready to Market
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SHA	Secure Hash Algorithm
SRK	Storage Root Key
STF	Self Test Failed
TA	Time Authority
TBB	Threading Building Blocks
TCG	Trusted Computing Group
TCV	Tick Count Value
TIR	Tick Increment Rate
TIS	TPM Interface Specification
TNC	Trusted Network Connect
TOE	Target of Evaluation
TOS	Trusted Operating System
TPCA	Trusted Platform Computing Alliance
TPM	Trusted Platform Module
TPME	Trusted Platform Module Entity
TSC	Tick Stamp Counter
TSC_	TPM Software Connection, when used as a command prefix
TSN	Tick Session Name
TSR	Tick Stamp Reset
TSRB	TickStampReset for blob
TSS	TCG Software Stack
TTP	Trusted Third Party/Time-Triggered Protocol
TS	Tick Stamp
UTC	Universal Time Clock
VPN	Virtual Private Network

4. Structures and Formats

Start of informative comment

The following structures and formats describe the interoperable areas of ISO/IEC 11889. There is no requirement that internal storage or memory representations of data must follow these structures. These requirements are in place only during the movement of data from a TPM to some other entity.

End of informative comment

4.1 Representation of Information

4.1.1 Endness of Structures

Each structure MUST use big endian bit ordering, which follows the Internet standard and requires that the low-order bit appear to the far right of a word, buffer, wire format, or other area and the high-order bit appear to the far left.

4.1.2 Byte Packing

All structures MUST be packed on a byte boundary.

4.1.3 Lengths

The “Byte” is the unit of length when the length of a parameter is specified.

4.1.4 Structure Definitions

This is not a MIDL compatible specification. The syntax of a structure definition is just a hint. It should be used with Description column to determine the nature of a structure member.

4.2 Defines

Start of informative comment

These definitions are in use to make a consistent use of values throughout the structure definitions.

Let x^* denote the usual C-style pointers, e.g. a reference to an object or array of objects of type x . Let $x[]$ denote an array with elements of type x . If a number of elements is explicitly given, like `BYTE[16]`, that means a precise number of elements (16, for example) of type `BYTE`. If no explicit number of elements is given, it indicates a variable-sized number of elements, where the number is given by some other means, e.g. another parameter.

End of informative comment

4.2.1 Basic data types

Parameters

Table 1: Basic data type parameters

Typedef	Name	Description
unsigned char	BYTE	Basic byte used to transmit all character fields.
unsigned char	BOOL	TRUE/FALSE field. TRUE = 0x01, FALSE = 0x00
unsigned short	UINT16	16-bit field. The definition in different architectures may need to specify 16 bits instead of the short definition
unsigned long	UINT32	32-bit field. The definition in different architectures may need to specify 32 bits instead of the long definition

4.2.2 Boolean types

Table 2: Boolean types

Name	Value	Description
TRUE	0x01	Assertion
FALSE	0x00	Contradiction

4.2.3 Helper redefinitions

The following definitions are to make the definitions more explicit and easier to read.

Parameters

Table 3: Helper redefinition parameters

Typedef	Name	Description
BYTE	TPM_AUTH_DATA_USAGE	Indicates the conditions where it is required that authorization be presented.
BYTE	TPM_PAYLOAD_TYPE	The information as to what the payload is in an encrypted structure
BYTE	TPM_VERSION_BYTE	The version info breakdown
BYTE	TPM_DA_STATE	The state of the dictionary attack mitigation logic
UINT16	TPM_TAG	The request or response authorization type.
UINT16	TPM_PROTOCOL_ID	The protocol in use.
UINT16	TPM_STARTUP_TYPE	Indicates the start state.
UINT16	TPM_ENC_SCHEME	The definition of the encryption scheme.
UINT16	TPM_SIG_SCHEME	The definition of the signature scheme.

Typedef	Name	Description
UINT16	TPM_MIGRATE_SCHEME	The definition of the migration scheme
UINT16	TPM_PHYSICAL_PRESENCE	Sets the state of the physical presence mechanism.
UINT16	TPM_ENTITY_TYPE	Indicates the types of entity that are supported by the TPM.
UINT16	TPM_KEY_USAGE	Indicates the permitted usage of the key.
UINT16	TPM_EK_TYPE	The type of asymmetric encrypted structure in use by the endorsement key
UINT16	TPM_STRUCTURE_TAG	The tag for the structure
UINT16	TPM_PLATFORM_SPECIFIC	The platform specific spec to which the information relates to
UINT32	TPM_COMMAND_CODE	The command ordinal.
UINT32	TPM_CAPABILITY_AREA	Identifies a TPM capability area.
UINT32	TPM_KEY_FLAGS	Indicates information regarding a key.
UINT32	TPM_ALGORITHM_ID	Indicates the type of algorithm.
UINT32	TPM_MODIFIER_INDICATOR	The locality modifier
UINT32	TPM_ACTUAL_COUNT	The actual number of a counter.
UINT32	TPM_TRANSPORT_ATTRIBUTES	Attributes that define what options are in use for a transport session
UINT32	TPM_AUTHHANDLE	Handle to an authorization session
UINT32	TPM_DIRINDEX	Index to a DIR register
UINT32	TPM_KEY_HANDLE	The area where a key is held assigned by the TPM.
UINT32	TPM_PCRINDEX	Index to a PCR register
UINT32	TPM_RESULT	The return code from a function
UINT32	TPM_RESOURCE_TYPE	The types of resources that a TPM may have using internal resources
UINT32	TPM_KEY_CONTROL	Allows for controlling of the key when loaded and how to handle TPM_Startup issues
UINT32	TPM_NV_INDEX	The index into the NV storage area
UINT32	TPM_FAMILY_ID	The family ID. Families ID's are automatically assigned a sequence number by the TPM. A trusted process can set the FamilyID value in an individual row to zero, which invalidates that row. The family ID resets to zero on each change of TPM Owner.
UINT32	TPM_FAMILY_VERIFICATION	A value used as a label for the most recent verification of this family. Set to zero when not in use.
UINT32	TPM_STARTUP_EFFECTS	How the TPM handles var
UINT32	TPM_SYM_MODE	The mode of a symmetric encryption
UINT32	TPM_FAMILY_FLAGS	The family flags
UINT32	TPM_DELEGATE_INDEX	The index value for the delegate NV table
UINT32	TPM_CMK_DELEGATE	The restrictions placed on delegation of CMK commands
UINT32	TPM_COUNT_ID	The ID value of a monotonic counter
UINT32	TPM_REEDIT_COMMAND	A command to execute
UINT32	TPM_TRANSHANDLE	A transport session handle
UINT32	TPM_HANDLE	A generic handle could be key, transport etc.
UINT32	TPM_FAMILY_OPERATION	What operation is happening

4.2.4 Vendor specific

Start of informative comment

For all items that can specify an individual algorithm, protocol or item ISO/IEC 11889 allows for vendor specific selections. The mechanism to specify a vendor specific mechanism is to set the high bit of the identifier on.

End of informative comment

The following defines allow for the quick specification of a vendor specific item.

Parameters

Table 4: Vendor specific parameters

Name	Value
TPM_Vendor_Specific32	0x00000400
TPM_Vendor_Specific8	0x80

5. Structure Tags

Start of informative comment

There have been some indications that knowing what structure is in use would be valuable information in each structure. This new tag will be in each new structure that the TPM defines.

The upper nibble of the value designates the purview of the structure tag. 0 is used for TPM structures, 1 for platforms, and 2-F are reserved.

End of informative comment

5.1 TPM_STRUCTURE_TAG

The upper nibble of the value MUST be 0 for all TPM structures.

TPM_StructureTags

Table 5: TPM_StructureTag

Name	Value	Structure
TPM_TAG_CONTEXTBLOB	0x0001	TPM_CONTEXT_BLOB
TPM_TAG_CONTEXT_SENSITIVE	0x0002	TPM_CONTEXT_SENSITIVE
TPM_TAG_CONTEXTPOINTER	0x0003	TPM_CONTEXT_POINTER
TPM_TAG_CONTEXTLIST	0x0004	TPM_CONTEXT_LIST
TPM_TAG_SIGNINFO	0x0005	TPM_SIGN_INFO
TPM_TAG_PCR_INFO_LONG	0x0006	TPM_PCR_INFO_LONG
TPM_TAG_PERSISTENT_FLAGS	0x0007	TPM_PERMANENT_FLAGS
TPM_TAG_VOLATILE_FLAGS	0x0008	TPM_VOLATILE_FLAGS
TPM_TAG_PERSISTENT_DATA	0x0009	TPM_PERSISTENT_DATA
TPM_TAG_VOLATILE_DATA	0x000A	TPM_VOLATILE_DATA
TPM_TAG_SV_DATA	0x000B	TPM_SV_DATA
TPM_TAG_EK_BLOB	0x000C	TPM_EK_BLOB
TPM_TAG_EK_BLOB_AUTH	0x000D	TPM_EK_BLOB_AUTH
TPM_TAG_COUNTER_VALUE	0x000E	TPM_COUNTER_VALUE
TPM_TAG_TRANSPORT_INTERNAL	0x000F	TPM_TRANSPORT_INTERNAL
TPM_TAG_TRANSPORT_LOG_IN	0x0010	TPM_TRANSPORT_LOG_IN
TPM_TAG_TRANSPORT_LOG_OUT	0x0011	TPM_TRANSPORT_LOG_OUT
TPM_TAG_AUDIT_EVENT_IN	0x0012	TPM_AUDIT_EVENT_IN
TPM_TAG_AUDIT_EVENT_OUT	0x0013	TPM_AUDIT_EVENT_OUT
TPM_TAG_CURRENT_TICKS	0x0014	TPM_CURRENT_TICKS
TPM_TAG_KEY	0x0015	TPM_KEY
TPM_TAG_STORED_DATA12	0x0016	TPM_STORED_DATA12
TPM_TAG_NV_ATTRIBUTES	0x0017	TPM_NV_ATTRIBUTES
TPM_TAG_NV_DATA_PUBLIC	0x0018	TPM_NV_DATA_PUBLIC
TPM_TAG_NV_DATA_SENSITIVE	0x0019	TPM_NV_DATA_SENSITIVE
TPM_TAG_DELEGATIONS	0x001A	TPM_DELEGATIONS
TPM_TAG_DELEGATE_PUBLIC	0x001B	TPM_DELEGATE_PUBLIC
TPM_TAG_DELEGATE_TABLE_ROW	0x001C	TPM_DELEGATE_TABLE_ROW
TPM_TAG_TRANSPORT_AUTH	0x001D	TPM_TRANSPORT_AUTH
TPM_TAG_TRANSPORT_PUBLIC	0x001E	TPM_TRANSPORT_PUBLIC
TPM_TAG_PERMANENT_FLAGS	0x001F	TPM_PERMANENT_FLAGS
TPM_TAG_STCLEAR_FLAGS	0x0020	TPM_STCLEAR_FLAGS
TPM_TAG_STANY_FLAGS	0x0021	TPM_STANY_FLAGS
TPM_TAG_PERMANENT_DATA	0x0022	TPM_PERMANENT_DATA
TPM_TAG_STCLEAR_DATA	0x0023	TPM_STCLEAR_DATA
TPM_TAG_STANY_DATA	0x0024	TPM_STANY_DATA

Name	Value	Structure
TPM_TAG_FAMILY_TABLE_ENTRY	0X0025	TPM_FAMILY_TABLE_ENTRY
TPM_TAG_DELEGATE_SENSITIVE	0X0026	TPM_DELEGATE_SENSITIVE
TPM_TAG_DELG_KEY_BLOB	0X0027	TPM_DELG_KEY_BLOB
TPM_TAG_KEY12	0x0028	TPM_KEY12
TPM_TAG_CERTIFY_INFO2	0X0029	TPM_CERTIFY_INFO2
TPM_TAG_DELEGATE_OWNER_BLOB	0X002A	TPM_DELEGATE_OWNER_BLOB
TPM_TAG_EK_BLOB_ACTIVATE	0X002B	TPM_EK_BLOB_ACTIVATE
TPM_TAG_DAA_BLOB	0X002C	TPM_DAA_BLOB
TPM_TAG_DAA_CONTEXT	0X002D	TPM_DAA_CONTEXT
TPM_TAG_DAA_ENFORCE	0X002E	TPM_DAA_ENFORCE
TPM_TAG_DAA_ISSUER	0X002F	TPM_DAA_ISSUER
TPM_TAG_CAP_VERSION_INFO	0X0030	TPM_CAP_VERSION_INFO
TPM_TAG_DAA_SENSITIVE	0X0031	TPM_DAA_SENSITIVE
TPM_TAG_DAA_TPM	0X0032	TPM_DAA_TPM
TPM_TAG_CMK_MIGAUTH	0X0033	TPM_CMK_MIGAUTH
TPM_TAG_CMK_SIGTICKET	0X0034	TPM_CMK_SIGTICKET
TPM_TAG_CMK_MA_APPROVAL	0X0035	TPM_CMK_MA_APPROVAL
TPM_TAG_QUOTE_INFO2	0X0036	TPM_QUOTE_INFO2
TPM_TAG_DA_INFO	0x0037	TPM_DA_INFO
TPM_TAG_DA_INFO_LIMITED	0x0038	TPM_DA_INFO_LIMITED
TPM_TAG_DA_ACTION_TYPE	0x0039	TPM_DA_ACTION_TYPE

6. Types

6.1 TPM_RESOURCE_TYPE

TPM_ResourceTypes

Table 6: TPM_ResourceTypes

Name	Value	Description
TPM_RT_KEY	0x00000001	The handle is a key handle and is the result of a LoadKey type operation
TPM_RT_AUTH	0x00000002	The handle is an authorization handle. Auth handles come from TPM_OIAP, TPM_OSAP and TPM_DSAP
TPM_RT_HASH	0x00000003	Reserved for hashes
TPM_RT_TRANS	0x00000004	The handle is for a transport session. Transport handles come from TPM_EstablishTransport
TPM_RT_CONTEXT	0x00000005	Resource wrapped and held outside the TPM using the context save/restore commands
TPM_RT_COUNTER	0x00000006	Reserved for counters
TPM_RT_DELEGATE	0x00000007	The handle is for a delegate row. These are the internal rows held in NV storage by the TPM
TPM_RT_DAA_TPM	0x00000008	The value is a DAA TPM specific blob
TPM_RT_DAA_V0	0x00000009	The value is a DAA V0 parameter
TPM_RT_DAA_V1	0x0000000A	The value is a DAA V1 parameter

6.2 TPM_PAYLOAD_TYPE

Start of informative comment

This structure specifies the type of payload in various messages.

The payload may indicate whether the key is a CMK, and the CMK type. The distinction was put here rather than in TPM_KEY_USAGE:

for backward compatibility

because some commands only see the TPM_STORE_ASYMKEY, not the entire TPM_KEY

End of informative comment

TPM_PAYLOAD_TYPE Values

Table 7: TPM_PAYLOAD_TYPE values

Value	Name	Comments
0x01	TPM_PT_ASYM	The entity is an asymmetric key
0x02	TPM_PT_BIND	The entity is bound data
0x03	TPM_PT_MIGRATE	The entity is a migration blob
0x04	TPM_PT_MAINT	The entity is a maintenance blob
0x05	TPM_PT_SEAL	The entity is sealed data
0x06	TPM_PT_MIGRATE_RESTRICTED	The entity is a restricted-migration asymmetric key
0x07	TPM_PT_MIGRATE_EXTERNAL	The entity is a external migratable key
0x08	TPM_PT_CMK_MIGRATE	The entity is a CMK migratable blob
0x09 – 0x7F		Reserved for future use by TPM
0x80 – 0xFF		Vendor specific payloads

6.3 TPM_ENTITY_TYPE

Start of informative comment

This specifies the types of entity and ADIP encryption schemes that are supported by the TPM.

The LSB is used to indicate the entity type. The MSB is used to indicate the ADIP encryption scheme when applicable.

For compatibility with TPM 1.1, this mapping is maintained:

0x0001 specifies a keyHandle entity with XOR encryption

0x0002 specifies an owner entity with XOR encryption

0x0003 specifies some data entity with XOR encryption

0x0004 specifies the SRK entity with XOR encryption

0x0005 specifies a key entity with XOR encryption

The method of incrementing the symmetric key counter value is different from that used by some standard crypto libraries (e.g. openssl, Java JCE) that increment the entire counter value. TPM users should be aware of this to avoid errors when the counter wraps.

End of informative comment

1. When the entity is not being used for ADIP encryption, the MSB MUST be 0x00.

TPM_ENTITY_TYPE LSB Values

Table 8: TPM_ENTITY_TYPE LSB Values

Value	Entity Name	Key Handle	Comments
0x01	TPM_ET_KEYHANDLE		The entity is a keyHandle or key
0x02	TPM_ET_OWNER	0x40000001	The entity is the TPM Owner
0x03	TPM_ET_DATA		The entity is some data
0x04	TPM_ET_SRK	0x40000000	The entity is the SRK
0x05	TPM_ET_KEY		The entity is a key or keyHandle
0x06	TPM_ET_REVOKE	0x40000002	The entity is the RevokeTrust value
0x07	TPM_ET_DEL_OWNER_BLOB		The entity is a delegate owner blob
0x08	TPM_ET_DEL_ROW		The entity is a delegate row
0x09	TPM_ET_DEL_KEY_BLOB		The entity is a delegate key blob
0x0A	TPM_ET_COUNTER		The entity is a counter
0x0B	TPM_ET_NV		The entity is a NV index
0x0C	TPM_ET_OPERATOR		The entity is the operator
0x40	TPM_ET_RESERVED_HANDLE		Reserved. This value avoids collisions with the handle MSB setting.

TPM_ENTITY_TYPE MSB Values

Table 9: TPM_ENTITY_TYPE MSB Values

Value	Algorithm	ADIP encryption scheme
0x00	TPM_ET_XOR	XOR
0x06	TPM_ET_AES_CTR	AES 128 bits in CTR mode

6.4 Handles

Start of informative comment

Handles provides pointers to TPM internal resources. Handles should provide the ability to locate an entity without collision. When handles are used, the TPM must be able to unambiguously determine the entity type.

Handles are 32 bit values. To enable ease of use in handles and to assist in internal use of handles the TPM will use the following rules when creating the handle.

The three least significant bytes (LSB) of the handle contain whatever entropy the TPM needs to provide collision avoidance. The most significant byte (MSB) may also be included.

Counter handles need not provide collision avoidance.

Reserved key handles

Certain TPM entities have handles that point specifically to them, like the SRK. These values always use the MSB of 0x40. This is a reserved key handle value and all special handles will use the 0x40 prefix.

Handle collisions

The TPM provides good, but not foolproof protection against handle collisions. If system or application software detects a collision that is problematic, the software should evict the resource, and re-submit the command.

End of informative comment

1. The TPM MUST generate key, authorization session, transport session, and DAA handles and MAY generate counter handles as follows:
 - a. The three LSB of the handle MUST and the MSB MAY contain the collision resistance values. The TPM MUST provide protection against handle collision. The TPM MUST implement one of the following:
 - b. The three LSB of the handle MUST and the MSB MAY be generated randomly. The TPM MUST ensure that no currently loaded entity of the same type has the same handle.
 - c. The three LSB of the handle MUST be generated from a monotonic counter. The monotonic counter value MUST NOT reset on TPM startup, but may wrap over the life of the TPM.
 - d. The MSB MAY be a value that does not contribute to collision resistance.
2. A key handle MUST NOT have the reserved value 0x40 in the MSB.
3. The TPM MAY use the counter index as the monotonic counter handle.
4. Handles are not required to be globally unique between entity groups (key, authorization session, transport session, and DAA).
5. For example, a newly generated authorization handle MAY have the same value as a loaded key handle.

6.4.1 Reserved Key Handles

Start of informative comment

The reserved key handles. These values specify specific keys or specific actions for the TPM.

TPM_KH_TRANSPORT indicates to TPM_EstablishTransport that there is no encryption key, and that the “secret” wrapped parameters are actually passed unencrypted.

End of informative comment

1. All reserved key handles MUST start with 0x40.
2. By default, when an ordinal input parameter specifies a TPM_KEY_HANDLE, a TPM generated key handle or TPM_KH_SRK can be used, but a reserved key handle other than TPM_KH_SRK can never be used.
3. The actions may further restrict use of any key. For example, TPM_KH_SRK cannot be used for TPM_Sign because it is not a signing key.
4. When an ordinal input parameter specifies a TPM_KEY_HANDLE, a reserved key handle can be used if explicitly allowed by the ordinal actions.
5. For example, TPM_CreateWrapKey or TPM_GetPubKey can use TPM_KH_SRK but not TPM_KH_EK by default.
6. For example, TPM_OwnerReadInternalPub can use TPM_KH_EK because it is explicitly allowed by the actions.

Key Handle Values

Table 10: Key Handle Values

Key Handle	Handle Name	Comments
0x40000000	TPM_KH_SRK	The handle points to the SRK
0x40000001	TPM_KH_OWNER	The handle points to the TPM Owner
0x40000002	TPM_KH_REVOKE	The handle points to the RevokeTrust value
0x40000003	TPM_KH_TRANSPORT	The handle points to the TPM_EstablishTransport static authorization
0x40000004	TPM_KH_OPERATOR	The handle points to the Operator auth
0x40000005	TPM_KH_ADMIN	The handle points to the delegation administration auth
0x40000006	TPM_KH_EK	The handle points to the PUBEK, only usable with TPM_OwnerReadInternalPub

6.5 TPM_STARTUP_TYPE

Start of informative comment

To specify what type of startup is occurring.

End of informative comment

TPM_STARTUP_TYPE Values

Table 11: TPM_STARTUP_TYPE Values

Value	Event Name	Comments
0x0001	TPM_ST_CLEAR	The TPM is starting up from a clean state
0x0002	TPM_ST_STATE	The TPM is starting up from a saved state
0x0003	TPM_ST_DEACTIVATED	The TPM is to startup and set the deactivated flag to TRUE

6.6 TPM_STARTUP_EFFECTS

Start of Informative comment

This structure lists for the various resources and sessions on a TPM the affect that TPM_Startup has on the values.

There are three ST_STATE options for keys (restore all, restore non-volatile, or restore none) and two ST_CLEAR options (restore non-volatile or restore none). As bit 4 was insufficient to describe the possibilities, it is deprecated. Software should use TPM_CAP_KEY_HANDLE to determine which keys are loaded after TPM_Startup.

End of informative comment

Types of Startup

Table 12: Types of Startup

Bit position	Name	Description
31-9		No information and MUST be FALSE
8		TPM_RT_DAA_TPM resources are initialized by TPM_Startup(ST_STATE)
7		TPM_Startup has no effect on auditDigest
6		auditDigest is set to all zeros on TPM_Startup(ST_CLEAR) but not on other types of TPM_Startup
5		auditDigest is set to all zeros on TPM_Startup(any)
4		Deprecated, as the meaning was subject to interpretation. (Was:TPM_RT_KEY resources are initialized by TPM_Startup(ST_ANY))
3		TPM_RT_AUTH resources are initialized by TPM_Startup(ST_STATE)
2		TPM_RT_HASH resources are initialized by TPM_Startup(ST_STATE)
1		TPM_RT_TRANS resources are initialized by TPM_Startup(ST_STATE)
0		TPM_RT_CONTEXT session (but not key) resources are initialized by TPM_Startup(ST_STATE)

6.7 TPM_PROTOCOL_ID

Start of informative comment

This value identifies the protocol in use.

End of informative comment

TPM_PROTOCOL_ID Values

Table 13: TPM_PROTOCOL_ID Values

Value	Event Name	Comments
0x0001	TPM_PID_OIAP	The OIAP protocol.
0x0002	TPM_PID_OSAP	The OSAP protocol.
0x0003	TPM_PID_ADIP	The ADIP protocol.
0X0004	TPM_PID_ADCP	The ADCP protocol.
0X0005	TPM_PID_OWNER	The protocol for taking ownership of a TPM.
0x0006	TPM_PID_DSAP	The DSAP protocol
0x0007	TPM_PID_TRANSPORT	The transport protocol

6.8 TPM_ALGORITHM_ID

Start of informative comment

This table defines the types of algorithms that may be supported by the TPM.

End of informative comment

TPM_ALGORITHM_ID values

Table 14: TPM_ALGORITHM_ID values

Value	Name	Description
0x00000001	TPM_ALG_RSA	The RSA algorithm.
0x00000002	reserved	(was the DES algorithm)
0x00000003	reserved	(was the 3DES algorithm in EDE mode)
0x00000004	TPM_ALG_SHA	The SHA1 algorithm
0x00000005	TPM_ALG_HMAC	The RFC 2104 HMAC algorithm
0x00000006	TPM_ALG_AES	The AES algorithm, key size 128
0x00000007	TPM_ALG_MGF1	The XOR algorithm using MGF1 to create a string the size of the encrypted block
0x00000008	Reserved	Was AES, key size 192
0x00000009	Reserved	Was AES, key size 256
0x0000000A	TPM_ALG_XOR	XOR using the rolling nonces

Description

The TPM MUST support the algorithms TPM_ALG_RSA, TPM_ALG_SHA, TPM_ALG_HMAC, and TPM_ALG_MGF1

6.9 TPM_PHYSICAL_PRESENCE

Table 15: TPM_PHYSICAL_PRESENCE parameters

Name	Value	Description
TPM_PHYSICAL_PRESENCE_HW_DISABLE	0x0200h	Sets the physicalPresenceHwEnable to FALSE
TPM_PHYSICAL_PRESENCE_CMD_DISABLE	0x0100h	Sets the physicalPresenceCmdEnable to FALSE
TPM_PHYSICAL_PRESENCE_LIFETIME_LOCK	0x0080h	Sets the physicalPresenceLifetimeLock to TRUE
TPM_PHYSICAL_PRESENCE_HW_ENABLE	0x0040h	Sets the physicalPresenceHwEnable to TRUE
TPM_PHYSICAL_PRESENCE_CMD_ENABLE	0x0020h	Sets the physicalPresenceCmdEnable to TRUE
TPM_PHYSICAL_PRESENCE_NOTPRESENT	0x0010h	Sets PhysicalPresence = FALSE
TPM_PHYSICAL_PRESENCE_PRESENT	0x0008h	Sets PhysicalPresence = TRUE
TPM_PHYSICAL_PRESENCE_LOCK	0x0004h	Sets PhysicalPresenceLock = TRUE

6.10 TPM_MIGRATE_SCHEME

Start of informative comment

The scheme indicates how the StartMigrate command should handle the migration of the encrypted blob.

End of informative comment

TPM_MIGRATE_SCHEME values

Table 16: TPM_MIGRATE_SCHEME values

Name	Value	Description
TPM_MS_MIGRATE	0x0001	A public key that can be used with all TPM migration commands other than 'ReWrap' mode.
TPM_MS_REWRAP	0x0002	A public key that can be used for the ReWrap mode of TPM_CreateMigrationBlob.
TPM_MS_MAINT	0x0003	A public key that can be used for the Maintenance commands
TPM_MS_RESTRICT_MIGRATE	0x0004	The key is to be migrated to a Migration Authority.
TPM_MS_RESTRICT_APPROVE	0x0005	The key is to be migrated to an entity approved by a Migration Authority using double wrapping

6.11 TPM_EK_TYPE

Start of informative comment

This structure indicates what type of information that the EK is dealing with.

End of informative comment

Table 17: TPM_EK_TYPE parameters

Name	Value	Description
TPM_EK_TYPE_ACTIVATE	0x0001	The blob MUST be TPM_EK_BLOB_ACTIVATE
TPM_EK_TYPE_AUTH	0x0002	The blob MUST be TPM_EK_BLOB_AUTH

6.12 TPM_PLATFORM_SPECIFIC

Start of informative comment

This enumerated type indicates the platform specific spec that the information relates to.

End of informative comment

Table 18: TPM_PLATFORM_SPECIFIC parameters

Name	Value	Description
TPM_PS_PC_11	0x0001	PC Specific version 1.1
TPM_PS_PC_12	0x0002	PC Specific version 1.2
TPM_PS_PDA_12	0x0003	PDA Specific version 1.2
TPM_PS_Server_12	0x0004	Server Specific version 1.2
TPM_PS_Mobile_12	0x0005	Mobile Specific version 1.2

7. Basic Structures

7.1 TPM_STRUCT_VER

Start of informative comment

This indicates the version of the structure.

Version 1.2 deprecates the use of this structure in all other structures. The structure is not deprecated as many of the structures that contain this structure are not deprecated.

The rationale behind keeping this structure and adding the new version structure is that in version 1.1 this structure was in use for two purposes. The first was to indicate the structure version, and in that mode the revMajor and revMinor were supposed to be set to 0. The second use was in TPM_GetCapability and the structure would then return the correct revMajor and revMinor. This use model caused problems in keeping track of when the revs were or were not set and how software used the information. Version 1.2 went to structure tags. Some structures did not change and the TPM_STRUCT_VER is still in use. To avoid the problems from 1.1, this structure now is a fixed value and only remains for backwards compatibility. Structure versioning comes from the tag on the structure, and the TPM_GetCapability response for TPM versioning uses TPM_VERSION.

End of informative comment

Definition

```
typedef struct tdTPM_STRUCT_VER {
    BYTE major;
    BYTE minor;
    BYTE revMajor;
    BYTE revMinor;
} TPM_STRUCT_VER;
```

Parameters

Table 19: TPM_STRUCT_VER parameters

Type	Name	Description
BYTE	major	This SHALL indicate the major version of the structure. MUST be 0x01
BYTE	minor	This SHALL indicate the minor version of the structure. MUST be 0x01
BYTE	revMajor	This MUST be 0x00 on output, ignored on input
BYTE	revMinor	This MUST be 0x00 on output, ignored on input

Descriptions

1. Provides the version of the structure
2. The TPM SHALL inspect the major and minor fields to determine if the TPM can properly interpret the structure.
 - a. On error, the TPM MUST return TPM_BAD_VERSION.
 - b. The TPM MUST ignore the revMajor and revMinor fields on input.

7.2 TPM_VERSION_BYTE

Start of Informative comment

Allocating a byte for the version information is wasteful of space. The current allocation does not provide sufficient resolution to indicate completely the version of the TPM. To allow for backwards compatibility the size of the structure does not change from 1.1.

To enable minor version numbers with 2-digit resolution, the byte representing a version splits into two BCD encoded nibbles. The ordering of the low and high order provides backwards compatibility with existing numbering.

An example of an implementation of this is; a version of 1.23 would have the value 2 in bit positions 3-0 and the value 3 in bit positions 7-4.

End of informative comment

TPM_VERSION_BYTE is a byte. The byte is broken up according to the following rule

Table 20: TPM_VERSION_BYTE rule

Bit position	Name	Description
7-4	leastSigVer	Least significant nibble of the minor version. MUST be values within the range of 0000-1001
3-0	mostSigVer	Most significant nibble of the minor version. MUST be values within the range of 0000-1001

7.3 TPM_VERSION

Start of informative comment

This structure provides information relative the version of the TPM. This structure should only be in use by TPM_GetCapability to provide the information relative to the TPM.

End of informative comment

Definition

```
typedef struct tdTPM_VERSION {
    TPM_VERSION_BYTE major;
    TPM_VERSION_BYTE minor;
    BYTE revMajor;
    BYTE revMinor;
} TPM_VERSION;
```

Parameters

Table 21: TPM_VERSION parameters

Type	Name	Description
TPM_VERSION_BYTE	Major	This SHALL indicate the major version of the TPM, mostSigVer MUST be 0x01, leastSigVer MUST be 0x00
TPM_VERSION_BYTE	Minor	This SHALL indicate the minor version of the TPM, mostSigVer MUST be 0x01 or 0x02, leastSigVer MUST be 0x00
BYTE	revMajor	This SHALL be the value of the TPM_PERMANENT_DATA -> revMajor
BYTE	revMinor	This SHALL be the value of the TPM_PERMANENT_DATA -> revMinor

Descriptions

1. The major and minor fields indicate the specification version the TPM was designed for
2. The revMajor and revMinor fields indicate the manufacturer's revision of the TPM
 - a. Most challengers of the TPM MAY ignore the revMajor and revMinor fields

7.4 TPM_DIGEST

Start of informative comment

The digest value reports the result of a hash operation.

In version 1 the hash algorithm is SHA-1 with a resulting hash result being 20 bytes or 160 bits.

It is understood that algorithm agility is lost due to fixing the hash at 20 bytes and on SHA-1. The reason for fixing is due to the internal use of the digest. It is the AuthData values, it provides the secrets for the HMAC and the size of 20 bytes determines the values that can be stored and encrypted. For this reason, the size is fixed and any changes to this value require a new version of the specification.

End of informative comment

Definition

```
typedef struct tdTPM_DIGEST{
    BYTE[digestSize] digest;
} TPM_DIGEST;
```

Parameters

Table 22: TPM_DIGEST parameters

Type	Name	Description
BYTE	Digest	This SHALL be the actual digest information

Description

The digestSize parameter MUST indicate the block size of the algorithm and MUST be 20 or greater.

For all TPM v1 hash operations, the hash algorithm MUST be SHA-1 and the digestSize parameter is therefore equal to 20.

Redefinitions

Table 23: TPM_DIGEST redefinitions

Typedef	Name	Description
TPM_DIGEST	TPM_CHOSENID_HASH	This SHALL be the digest of the chosen identityLabel and privacyCA for a new TPM identity.
TPM_DIGEST	TPM_COMPOSITE_HASH	This SHALL be the hash of a list of PCR indexes and PCR values that a key or data is bound to.
TPM_DIGEST	TPM_DIRVALUE	This SHALL be the value of a DIR register
TPM_DIGEST	TPM_HMAC	
TPM_DIGEST	TPM_PCRVALUE	The value inside of the PCR
TPM_DIGEST	TPM_AUDITDIGEST	This SHALL be the value of the current internal audit state

7.4.1 Creating a PCR composite hash

The definition specifies the operation necessary to create TPM_COMPOSITE_HASH.

Action

1. The hashing MUST be done using the SHA-1 algorithm.
2. The input must be a valid TPM_PCR_SELECTION structure.
3. The process creates a TPM_PCR_COMPOSITE structure from the TPM_PCR_SELECTION structure and the PCR values to be hashed. If constructed by the TPM the values MUST come from the current PCR registers indicated by the PCR indices in the TPM_PCR_SELECTION structure.
4. The process then computes a SHA-1 digest of the TPM_PCR_COMPOSITE structure.
5. The output is the SHA-1 digest just computed.

7.5 TPM_NONCE

Start of informative comment

A nonce is a random value that provides protection from replay and other attacks. Many of the commands and protocols in ISO/IEC 11889 require a nonce. This structure provides a consistent view of what a nonce is.

End of informative comment

Definition

```
typedef struct tdTPM_NONCE{
    BYTE[20] nonce;
} TPM_NONCE;
```

Parameters

Table 24: TPM_NONCE parameters

Type	Name	Description
BYTE	Nonce	This SHALL be the 20 bytes of random data. When created by the TPM the value MUST be the next 20 bytes from the RNG.

Redefinitions

Table 25: TPM_NONCE redefinitions

Typedef	Name	Description
TPM_NONCE	TPM_DAA_TPM_SEED	This SHALL be a random value generated by a TPM immediately after the EK is installed in that TPM, whenever an EK is installed in that TPM
TPM_NONCE	TPM_DAA_CONTEXT_SEED	This SHALL be a random value

7.5.1 TPM_PROOF

Start of informative comment

A value of type TPM_PROOF is a value that the TPM uses internally to authenticate data stored externally. The authentication occurs when the TPM validates the integrity of the data loaded from some external source. Note that a TPM_PROOF value must never leave the TPM in cleartext form. The generation of the proof value occurs during TPM_TakeOwnership operation and the value is stored in a TPM_Shielded-Location.

End of informative comment

Definition

```
typedef struct tdTPM_PROOF{
    BYTE[20] proof;
} TPM_PROOF;
```

Parameters

Table 26: TPM_PROOF parameters

Type	Name	Description
BYTE	Nonce	This SHALL be the 20 bytes of known only to the TPM. The data is generated by the TPM upon execution of TPM_TakeOwnership using the TPM RNG.

7.6 TPM_AUTHDATA

Start of informative comment

The AuthData data is the information that is saved or passed to provide proof of ownership of an entity. For version 1 this area is always 20 bytes.

End of informative comment

Definition

```
typedef BYTE[20] tTPM_AUTHDATA;
```

Descriptions

When sending AuthData data to the TPM the TPM does not validate the decryption of the data. It is the responsibility of the entity owner to validate that the AuthData data was properly received by the TPM. This could be done by immediately attempting to open an authorization session.

The owner of the data can select any value for the data

Redefinitions

Table 27: TPM_AUTHDATA redefinitions

Typedef	Name	Description
TPM_AUTHDATA	TPM_SECRET	A secret plaintext value used in the authorization process.
TPM_AUTHDATA	TPM_ENCAUTH	A ciphertext (encrypted) version of AuthData data. The encryption mechanism depends on the context.

7.7 TPM_KEY_HANDLE_LIST

Start of informative comment

TPM_KEY_HANDLE_LIST is a structure used to describe the handles of all keys currently loaded into a TPM.

End of informative comment

Definition

```
typedef struct tdTPM_KEY_HANDLE_LIST {
    UINT16    loaded;
    [size_is(loaded)] TPM_KEY_HANDLE    handle[];
} TPM_KEY_HANDLE_LIST;
```

Parameters

Table 28: TPM_KEY_HANDLE_LIST parameters

Type	Name	Description
UINT16	loaded	The number of keys currently loaded in the TPM.
UINT32	handle	An array of handles, one for each key currently loaded in the TPM

Description

The order in which keys are reported is manufacturer-specific.

7.8 TPM_KEY_USAGE values

Start of informative comment

This table defines the types of keys that are possible. Each value defines for what operation the key can be used. Most key usages can be CMKs. See 6.2, TPM_PAYLOAD_TYPE.

Each key has a setting defining the encryption and signature scheme to use. The selection of a key usage value limits the choices of encryption and signature schemes.

End of informative comment

Table 29: TPM_KEY_USAGE values

Name	Value	Description
TPM_KEY_SIGNING	0x0010	This SHALL indicate a signing key. The [private] key SHALL be used for signing operations, only. This means that it MUST be a leaf of the Protected Storage key hierarchy.
TPM_KEY_STORAGE	0x0011	This SHALL indicate a storage key. The key SHALL be used to wrap and unwrap other keys in the Protected Storage hierarchy
TPM_KEY_IDENTITY	0x0012	This SHALL indicate an identity key. The key SHALL be used for operations that require a TPM identity, only.
TPM_KEY_AUTHCHANGE	0x0013	This SHALL indicate an ephemeral key that is in use during the ChangeAuthAsym process, only.
TPM_KEY_BIND	0x0014	This SHALL indicate a key that can be used for TPM_Bind and TPM_UnBind operations only.
TPM_KEY_LEGACY	0x0015	This SHALL indicate a key that can perform signing and binding operations. The key MAY be used for both signing and binding operations. The TPM_KEY_LEGACY key type is to allow for use by applications where both signing and encryption operations occur with the same key. The use of this key type is not recommended
TPM_KEY_MIGRATE	0x0016	This SHALL indicate a key in use for TPM_MigrateKey

7.8.1 Mandatory Key Usage Schemes

Start of Informative comment

For a given key usage type there are subset of valid encryption and signature schemes.

The method of incrementing the symmetric key counter value is different from that used by some standard crypto libraries (e.g. openssl, Java JCE) that increment the entire counter value. TPM users should be aware of this to avoid errors when the counter wraps.

End of informative comment

The key usage value for a key determines the encryption and / or signature schemes which MUST be used with that key. The table below maps the schemes defined by ISO/IEC 11889 to the defined key usage values.

Table 30: Mandatory Key Usage Schemes

Name	Allowed Encryption schemes	Allowed Signature Schemes
TPM_KEY_SIGNING	TPM_ES_NONE	TPM_SS_RSASSAPKCS1v15_SHA1 TPM_SS_RSASSAPKCS1v15_DER TPM_SS_RSASSAPKCS1v15_INFO
TPM_KEY_STORAGE	TPM_ES_RSAESOAEP_SHA1_MGF1	TPM_SS_NONE
TPM_KEY_IDENTITY	TPM_ES_NONE	TPM_SS_RSASSAPKCS1v15_SHA1
TPM_KEY_AUTHCHANGE	TPM_ES_RSAESOAEP_SHA1_MGF1	TPM_SS_NONE
TPM_KEY_BIND	TPM_ES_RSAESOAEP_SHA1_MGF1 TPM_ES_RSAESPKCSV15	TPM_SS_NONE
TPM_KEY_LEGACY	TPM_ES_RSAESOAEP_SHA1_MGF1 TPM_ES_RSAESPKCSV15	TPM_SS_RSASSAPKCS1v15_SHA1 TPM_SS_RSASSAPKCS1v15_DER
TPM_KEY_MIGRATE	TPM_ES_RSAESOAEP_SHA1_MGF1	TPM_SS_NONE

Where manufacturer specific schemes are used, the strength must be at least that listed in the above table for TPM_KEY_STORAGE, TPM_KEY_IDENTITY and TPM_KEY_AUTHCHANGE key types.

The TPM MUST check that the encryption scheme defined for use with the key is a valid scheme for the key type, as follows:

Table 31: Valid encryption schemes

Key algorithm	Approved schemes	TPM_ENC_SCHEME
TPM_ALG_RSA	TPM_ES_NONE	0x0001
TPM_ALG_RSA	TPM_ES_RSAESPKCSV15	0x0002
TPM_ALG_RSA	TPM_ES_RSAESOAEP_SHA1_MGF1	0x0003
TPM_ALG_AES	TPM_ES_SYM_CTR	0x0004
TPM_ALG_AES	TPM_ES_SYM_OFB	0x0005

The TPM MUST check that the signature scheme defined for use with the key is a valid scheme for the key type, as follows:

Table 32: Valid signature schemes

Key algorithm	Approved schemes	Scheme Value
TPM_ALG_RSA	TPM_SS_NONE	0x0001
	TPM_SS_RSASSAPKCS1v15_SHA1	0x0002
	TPM_SS_RSASSAPKCS1v15_DER	0x0003
	TPM_SS_RSASSAPKCS1v15_INFO	0x0004

7.9 TPM_AUTH_DATA_USAGE values

Start of informative comment

The indication to the TPM when authorization sessions for an entity are required. Future versions may allow for more complex decisions regarding AuthData checking.

Table 33: Combinations of TPM_AUTH_DATA_USAGE values

Tag	Ordinal Table	TPM_AUTH_DATA_USAGE	Action
AUTH1_COMMAND	AUTH1	TPM_AUTH_ALWAYS	Authorization HMAC is checked
AUTH1_COMMAND	AUTH1	TPM_AUTH_NEVER	Authorization HMAC is checked
AUTH1_COMMAND	AUTH1,RQU	TPM_AUTH_ALWAYS	Authorization HMAC is checked
AUTH1_COMMAND	AUTH1,RQU	TPM_AUTH_NEVER	Authorization HMAC is checked
RQU_COMMAND	AUTH1		Return TPM_BADTAG
RQU_COMMAND	AUTH1,RQU	TPM_AUTH_ALWAYS	Return TPM_AUTHFAIL
RQU_COMMAND	AUTH1,RQU	TPM_AUTH_NEVER	Allow command with no authorization check

The above table describes various combinations.

Lines 1-4 say that if an authorization HMAC is present and the ordinal table allows authorization, it is always checked. The TPM_AUTH_DATA_USAGE value is ignored.

Line 4 is often missed. That is, even if the ordinal table says that the command can be run without authorization and TPM_AUTH_DATA_USAGE says TPM_AUTH_NEVER, authorization can be present. If present, it is checked. TPM_AUTH_NEVER means that authorization may not be required for the key. It does not mean that authorization is not allowed.

Line 5 says that, if an authorization HMAC is not present, but the ordinal table says that authorization is required for the ordinal, TPM_BADTAG is returned. The TPM_AUTH_DATA_USAGE value is ignored. For example, TPM_CreateWrapKey always requires authorization, even if the key has TPM_AUTH_NEVER set.

Line 6 says that, if an authorization HMAC is not present, the ordinal table allows the command without the HMAC, but TPM_AUTH_ALWAYS is set, TPM_AUTHFAIL is returned. Even though the ordinal in general allows no authorization, the key used for this command requires authorization.

Line 7 says that, if an authorization HMAC is not present, the ordinal table allows the command without the HMAC, and TPM_AUTH_NEVER is set, the command is allowed to execute without authorization. The ordinal in general permits no authorization, and the key also permits no authorization.

End of informative comment

Table 34: TPM_AUTH_DATA_USAGE values

Name	Value	Description
TPM_AUTH_NEVER	0x00	This SHALL indicate that usage of the key without authorization is permitted.
TPM_AUTH_ALWAYS	0x01	This SHALL indicate that on each usage of the key the authorization MUST be performed.
TPM_AUTH_PRIV_USE_ONLY	0x03	This SHALL indicate that on commands that require the TPM to use the private portion of the key, the authorization MUST be performed. For commands that cause the TPM to read the public portion of the key, but not to use the private portion (e.g. TPM_GetPubKey), the authorization may be omitted.
		All other values are reserved for future use.