INTERNATIONAL STANDARD

ISO/IEC 11889-3

First edition
2009-05-15

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)

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.

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_APPROVALparameters 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

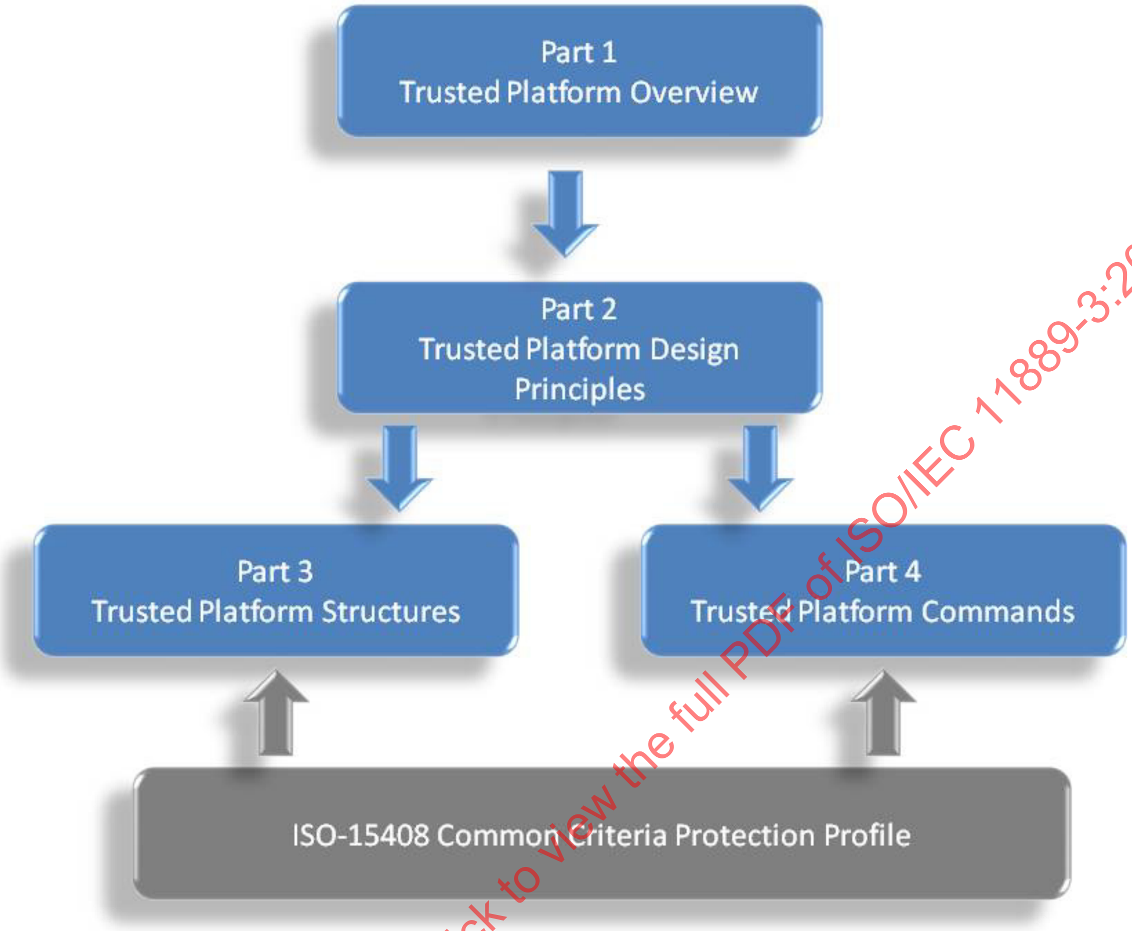
Part 1
Trusted Platform Overview
Part 2
Trusted Platform Design Principles
Part 3
Trusted Platform Structures
Part 4
Trusted Platform Commands
ISO-15408 Common Criteria Protection Profile

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_REDIT_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 redefinitons

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] tdTPM_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.

7.10 TPM_KEY_FLAGS

Start of informative comment

This table defines the meanings of the bits in a TPM_KEY_FLAGS structure, used in TPM_KEY and TPM_CERTIFY_INFO.

End of informative comment

TPM_KEY_FLAGS Values

Table 35: TPM_KEY_FLAGS Values

Name	Mask Value	Description
redirection	0x00000001	This mask value SHALL indicate the use of redirected output.
migratable	0x00000002	This mask value SHALL indicate that the key is migratable.
isVolatile	0x00000004	This mask value SHALL indicate that the key MUST be unloaded upon execution of the TPM_Startup(ST_Clear). This does not indicate that a nonvolatile key will remain loaded across TPM_Startup(ST_Clear) events.
pcrIgnoredOnRead	0x00000008	When TRUE the TPM MUST NOT check digestAtRelease or localityAtRelease for commands that use the public portion of the key like TPM_GetPubKey When FALSE the TPM MUST check digestAtRelease and localityAtRelease for commands that use the public portion of the key
migrateAuthority	0x00000010	When set indicates that the key is under control of a migration authority. The TPM MUST only allow the creation of a key with this flag in TPM_CMK_CreateKey

The value of TPM_KEY_FLAGS MUST be decomposed into individual mask values. The presence of a mask value SHALL have the effect described in the above table

On input, all undefined bits MUST be zero. The TPM MUST return an error if any undefined bit is set. On output, the TPM MUST set all undefined bits to zero.

7.11 TPM_CHANGEAUTH_VALIDATE

Start of informative comment

This structure provides an area that will stores the new AuthData data and the challenger's nonce.

End of informative comment

Definition

```
typedef struct tdTPM_CHANGEAUTH_VALIDATE {
   TPM_SECRET newAuthSecret;
   TPM_NONCE n1;
} TPM_CHANGEAUTH_VALIDATE;
```

Parameters

Table 36: TPM_CHANGEAUTH_VALIDATE parameters

Type	Name	Description
TPM_SECRET	newAuthSecret	This SHALL be the new AuthData data for the target entity
TPM_NONCE	n1	This SHOULD be a nonce, to enable the caller to verify that the target TPM is on-line.

7.12 TPM_MIGRATIONKEYAUTH

Start of informative comment

This structure provides the proof that the associated public key has TPM Owner AuthData to be a migration key.

End of informative comment

Definition

```
typedef struct tdTPM_MIGRATIONKEYAUTH{
   TPM_PUBKEY migrationKey;
   TPM_MIGRATE_SCHEME migrationScheme;
   TPM_DIGEST digest;
} TPM_MIGRATIONKEYAUTH;
```

Parameters

Table 37: TPM_MIGRATIONKEYAUTH parameters

Type	Name	Description
TPM_PUBKEY	migrationKey	This SHALL be the public key of the migration facility
TPM_MIGRATE_SCHEME	migrationScheme	This shall be the type of migration operation.
TPM_DIGEST	digest	This SHALL be the digest value of the concatenation of migration key, migration scheme and tpmProof

7.13 TPM_COUNTER_VALUE

Start of informative comment

This structure returns the counter value. For interoperability, the value size should be 4 bytes.

End of informative comment

Definition

```
typedef struct tdTPM_COUNTER_VALUE{
   TPM_STRUCTURE_TAG  tag;
   BYTE[4] label;
   TPM_ACTUAL_COUNT counter;
} TPM_COUNTER_VALUE;
```

Parameters

Table 38: TPM_COUNTER_VALUE parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_COUNTER_VALUE
BYTE	label	The label for the counter
TPM_ACTUAL_COUNT	counter	The 32-bit counter value.

7.14 TPM_SIGN_INFO Structure

Start of informative comment

This is an addition in 1.2 and is the structure signed for certain commands (e.g., TPM_ReleaseTransportSigned). Some commands have a structure specific to that command (e.g., TPM_Quote uses TPM_QUOTE_INFO) and do not use TPM_SIGN_INFO.

TPM_Sign uses this structure when the signature scheme is TPM_SS_RSASSAPKCS1v15_INFO.

End of informative comment

Definition

```
typedef struct tdTPM_SIGN_INFO {
   TPM_STRUCTURE_TAG tag;
   BYTE[4] fixed;
   TPM_NONCE replay;
   UINT32 dataLen;
   [size_is (dataLen)] BYTE[] data;
} TPM_SIGN_INFO;
```

Parameters

Table 39: TPM_SIGN_INFO Structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_SIGNINFO
BYTE	fixed	The ASCII text that identifies what function was performing the signing operation
TPM_NONCE	replay	Nonce provided by caller to prevent replay attacks
UINT32	dataLen	The length of the data area
BYTE	data	The data that is being signed

7.15 TPM_MSA_COMPOSITE

Start of informative comment

TPM_MSA_COMPOSITE contains an arbitrary number of digests of public keys belonging to Migration Authorities. An instance of TPM_MSA_COMPOSITE is incorporated into the migrationAuth value of a certified-migration-key (CMK), and any of the Migration Authorities specified in that instance is able to approve the migration of that certified-migration-key.

End of informative comment

Definition

```
typedef struct tdTPM_MSA_COMPOSITE {
   UINT32 MSAlist;
   TPM_DIGEST[] migAuthDigest[];
} TPM_MSA_COMPOSITE;
```

Parameters

Table 40: TPM_MSA_COMPOSITE parameters

Type	Name	Description
UINT32	MSAlist	The number of migAuthDigests. MSAlist MUST be one (1) or greater.
TPM_DIGEST[]	migAuthDigest[]	An arbitrary number of digests of public keys belonging to Migration Authorities.

TPMs MUST support TPM_MSA_COMPOSITE structures with MSAlist of four (4) or less, and MAY support larger values of MSAlist.

7.16 TPM_CMK_AUTH

Start of informative comment

The signed digest of TPM_CMK_AUTH is a ticket to prove that an entity with public key “migrationAuthority” has approved the public key “destination Key” as a migration destination for the key with public key “sourceKey”.

Normally the digest of TPM_CMK_AUTH is signed by the private key corresponding to “migrationAuthority”.

To reduce data size, TPM_CMK_AUTH contains just the digests of “migrationAuthority”, “destinationKey” and “sourceKey”.

End of informative comment

Definition

```
typedef struct tdTPM_CMK_AUTH{
   TPM_DIGEST migrationAuthorityDigest;
   TPM_DIGEST destinationKeyDigest;
   TPM_DIGEST sourceKeyDigest;
} TPM_CMK_AUTH;
```

Parameters

Table 41: TPM_CMK_AUTH parameters

Type	Name	Description
TPM_DIGEST	migrationAuthorityDigest	The digest of a public key belonging to a Migration Authority
TPM_DIGEST	destinationKeyDigest	The digest of a TPM_PUBKEY structure that is an approved destination key for the private key associated with “sourceKey”
TPM_DIGEST	sourceKeyDigest	The digest of a TPM_PUBKEY structure whose corresponding private key is approved by a Migration Authority to be migrated as a child to the destinationKey.

7.17 TPM_CMK_DELEGATE values

Start of informative comment

The bits of TPM_CMK_DELEGATE are flags that determine how the TPM responds to delegated requests to manipulate a certified-migration-key, a loaded key with payload type TPM_PT_MIGRATE_RESTRICTED or TPM_PT_MIGRATE_EXTERNAL.

End of informative comment

Table 42: TPM_CMK_DELEGATE values

Bit	Name	Description
31	TPM_CMK_DELEGATE_SIGNING	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_SIGNING
30	TPM_CMK_DELEGATE_STORAGE	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_STORAGE
29	TPM_CMK_DELEGATE_BIND	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_BIND
28	TPM_CMK_DELEGATE_LEGACY	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_LEGACY
27	TPM_CMK_DELEGATE_MIGRATE	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_MIGRATE
26:0	reserved	MUST be 0

The default value of TPM_CMK_Delegate is zero (0)

7.18 TPM_SELECT_SIZE

Start of informative comment

This structure provides the indication for the version and sizeOfSelect structure in TPM_GetCapability. Entities wishing to know if the TPM supports, for a specific version, a specific size fills in this structure and requests a TPM_GetCapability response from the TPM.

For instance, the entity would fill in version 1.1 and size 2. As 2 was the default size the TPM should return true. Filling in 1.1 and size 3, would return true or false depending on the capabilities of the TPM. For 1.2 the default size is 3 so all TPM's should support that size.

The real purpose of this structure is to see if the TPM supports an optional size for previous versions.

End of informative comment

Definition

```
typedef struct tdTPM_SELECT_SIZE {
   BYTE major;
   BYTE minor;
   UINT16 reqSize;
} TPM_SELECT_SIZE;
```

Parameters

Table 43: TPM_SELECT_SIZE parameters

Type	Name	Description
BYTE	Major	This SHALL indicate the major version of the TPM. This MUST be 0x01
BYTE	Minor	This SHALL indicate the minor version of the TPM. This MAY be 0x01 or 0x02
UINT16	reqSize	This SHALL indicate the value for a sizeOfSelect field in the TPM_SELECTION structure

7.19 TPM_CMK_MIGAUTH

Start of informative comment

Structure to keep track of the CMK migration authorization

End of informative comment

Definition

```
typedef struct tdTPM_CMK_MIGAUTH{
   TPM_STRUCTURE_TAG tag;
   TPM_DIGEST msaDigest;
   TPM_DIGEST pubKeyDigest;
} TPM_CMK_MIGAUTH;
```

Parameters

Table 44: TPM_CMK_MIGAUTH parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_CMK_MIGAUTH
TPM_DIGEST	msaDigest	The digest of a TPM_MSA_COMPOSITE structure containing the migration authority public key and parameters.
TPM_DIGEST	pubKeyDigest	The hash of the associated public key

7.20 TPM_CMK_SIGTICKET

Start of informative comment

Structure to keep track of the CMK migration authorization

End of informative comment

Definition

```
typedef struct tdTPM_CMK_SIGTICKET{
   TPM_STRUCTURE_TAG tag;
   TPM_DIGEST verKeyDigest;
   TPM_DIGEST signedData;
} TPM_CMK_SIGTICKET;
```

Parameters

Table 45: TPM_CMK_SIGTICKET parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_CMK_SIGTICKET
TPM_DIGEST	verKeyDigest	The hash of a TPM_PUBKEY structure containing the public key and parameters of the key that can verify the ticket
TPM_DIGEST	signedData	The ticket data

7.21 TPM_CMK_MA_APPROVAL

Start of informative comment

Structure to keep track of the CMK migration authorization

End of informative comment

Definition

```
typedef struct tdTPM_CMK_MA_APPROVAL{
   TPM_STRUCTURE_TAG tag;
   TPM_DIGEST migrationAuthorityDigest;
} TPM_CMK_MA_APPROVAL;
```

Parameters

Table 46: TPM_CMK_MA_APPROVALparameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_CMK_MA_APPROVAL
TPM_DIGEST	migrationAuthorityDigest	The hash of a TPM_MSA_COMPOSITE structure containing the hash of one or more migration authority public keys and parameters.

8. TPM_TAG (Command and Response Tags)

Start of informative comment

These tags indicate to the TPM the construction of the command either as input or as output. The AUTH indicates that there are one or more AuthData values that follow the command parameters.

End of informative comment

Table 47: TPM_TAG (Command and Response Tags)

Tag	Name	Description
0x00C1	TPM_TAG_RQU_COMMAND	A command with no authentication.
0x00C2	TPM_TAG_RQU_AUTH1_COMMAND	An authenticated command with one authentication handle
0x00C3	TPM_TAG_RQU_AUTH2_COMMAND	An authenticated command with two authentication handles
0x00C4	TPM_TAG_RSP_COMMAND	A response from a command with no authentication
0x00C5	TPM_TAG_RSP_AUTH1_COMMAND	An authenticated response with one authentication handle
0x00C6	TPM_TAG_RSP_AUTH2_COMMAND	An authenticated response with two authentication handles

9. Internal Data Held By TPM

Start of Informative comment

There are many flags and data fields that the TPM must manage to maintain the current state of the TPM. The areas under TPM control have different lifetimes. Some areas are permanent, some reset upon TPM_Startup(ST_CLEAR) and some reset upon TPM_Startup(ST_STATE).

Previously the data areas were not grouped exactly according to their reset capabilities. It has become necessary to properly group the areas into the three classifications.

Each field has defined mechanisms to allow the control of the field. The mechanism may require authorization or physical presence to properly authorize the management of the field.

End of informative comment

9.1 TPM_PERMANENT_FLAGS

Start of Informative comment

These flags maintain state information for the TPM. The values are not affected by any TPM_Startup command.

The TPM_SetCapability command indicating TPM_PF_READPUBEK can set readPubek either TRUE or FALSE. It has more capability than the deprecated TPM_DisablePubekRead, which can only set readPubek to FALSE.

If the optional TSC_PhysicalPresence command is not implemented, physicalPresenceHWEnable should be set by the TPM vendor.

If the TSC_PhysicalPresence command is implemented, physicalPresenceHWEnable and/or physicalPresenceCMDEnable should be set and physicalPresenceLifetimeLock should be set before the TPM platform is delivered to the user.

End of informative comment

```
typedef struct tdTPM_PERMANENT_FLAGS{
   TPM_STRUCTURE_TAG tag;
   BOOL disable;
   BOOL ownership;
   BOOL deactivated;
   BOOL readPubek;
   BOOL disableOwnerClear;
   BOOL allowMaintenance;
   BOOL physicalPresenceLifetimeLock;
   BOOL physicalPresenceHWEnable;
   BOOL physicalPresenceCMDEnable;
   BOOL CEKPUsed;
   BOOL TPMpost;
   BOOL TPMpostLock;
   BOOL FIPS;
   BOOL operator;
   BOOL enableRevokeEK;
   BOOL nvLocked;
   BOOL readSRKPub;
   BOOL tpmEstablished;
   BOOL maintenanceDone;
   BOOL disableFullDALogicInfo;
} TPM_PERMANENT_FLAGS;
```

Parameters

Table 48: TPM_PERMANENT_FLAGS parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_PERMANENT_FLAGS	
BOOL	disable	The state of the disable flag. The default state is TRUE	TPM_PF_DISABLE
BOOL	ownership	The ability to install an owner. The default state is TRUE.	TPM_PF_OWNERSHIP
BOOL	deactivated	The state of the inactive flag. The default state is TRUE.	TPM_PF_DEACTIVATED
BOOL	readPubek	The ability to read the PUBEK without owner AuthData. The default state is TRUE.	TPM_PF_READPUBEK

Type	Name	Description	Flag Name
BOOL	disableOwnerClear	Whether the owner authorized clear commands are active. The default state is FALSE.	TPM_PF_DISABLEOWNERCLEAR
BOOL	allowMaintenance	Whether the TPM Owner may create a maintenance archive. The default state is TRUE.	TPM_PF_ALLOWMAINTENANCE
BOOL	physicalPresenceLifetimeLock	This bit can only be set to TRUE; it cannot be set to FALSE except during the manufacturing process. FALSE: The state of either physicalPresenceHWEnable or physicalPresenceCMDEnable MAY be changed. (DEFAULT) TRUE: The state of either physicalPresenceHWEnable or physicalPresenceCMDEnable MUST NOT be changed for the life of the TPM.	TPM_PF_PHYSICALPRESENCELIFETIMELOCK
BOOL	physicalPresenceHWEnable	FALSE: Disable the hardware signal indicating physical presence. (DEFAULT) TRUE: Enables the hardware signal indicating physical presence.	TPM_PF_PHYSICALPRESENCEHWENABLE
BOOL	physicalPresenceCMDEnable	FALSE: Disable the command indicating physical presence. (DEFAULT) TRUE: Enables the command indicating physical presence.	TPM_PF_PHYSICALPRESENCECMDENABLE
BOOL	CEKPUsed	TRUE: The PRIVEK and PUBEK were created using TPM_CreateEndorsementKeyPair. FALSE: The PRIVEK and PUBEK were created using a manufacturer's process. NOTE: This flag has no default value as the key pair MUST be created by one or the other mechanism.	TPM_PF_CEKPUSED
BOOL	TPMpost	The meaning of this bit clarified in rev87. While actual use does not match the name, for backwards compatibility there is no change to the name. TRUE: After TPM_Startup, if there is a call to TPM_ContinueSelfTest the TPM MUST execute the actions of TPM_SelfTestFull FALSE: After TPM_Startup, if there is a call to TPM_ContinueSelfTest the TPM MUST execute the actions of TPM_ContinueSelfTest If the TPM supports the implicit invocation of TPM_ContinueSelftTest upon the use of an untested resource, the TPM MUST use the TPMPost flag to execute the actions of either TPM_ContinueSelfTest or TPM_SelfTestFull The TPM manufacturer sets this bit during TPM manufacturing and the bit is unchangeable after shipping the TPM The default state is FALSE	TPM_PF_TPMPOST

Type	Name	Description	Flag Name
BOOL	TPMpostLock	With the clarification of TPMPost TPMpostLock is now unnecessary. This flag is now deprecated	TPM_PF_TPMPOSTLOCK
BOOL	FIPS	TRUE: This TPM operates in FIPS mode FALSE: This TPM does NOT operate in FIPS mode	TPM_PF_FIPS
BOOL	operator	TRUE: The operator AuthData value is valid FALSE: the operator AuthData value is not set (DEFAULT)	TPM_PF_OPERATOR
BOOL	enableRevokeEK	TRUE: The TPM_RevokeTrust command is active FALSE: the TPM RevokeTrust command is disabled	TPM_PF_ENABLEREVOKEEK
BOOL	nvLocked	TRUE: All NV area authorization checks are active FALSE: No NV area checks are performed, except for maxNVWrites. FALSE is the default value	TPM_PF_NV_LOCKED
BOOL	readSRKPub	TRUE: GetPubKey will return the SRK pub key FALSE: GetPubKey will not return the SRK pub key Default is FALSE	TPM_PF_READSRKPUB
BOOL	tpmEstablished	TRUE: TPM_HASH_START has been executed at some time FALSE: TPM_HASH_START has not been executed at any time Default is FALSE. Reset to FALSE using TSC_ResetEstablishmentBit	TPM_PF_TPMESTABLISHED
BOOL	maintenanceDone	TRUE: A maintenance archive has been created for the current SRK	TPM_PF_MAINTENANCEDONE
BOOL	disableFullDALogicInfo	TRUE: The full dictionary attack TPM_GetCapability info is deactivated. The returned structure is TPM_DA_INFO_LIMITED. FALSE: The full dictionary attack TPM_GetCapability info is activated. The returned structure is TPM_DA_INFO. Default is FALSE.	TPM_PF_DISABLEFULLDALOGICINFO

Description

These values are permanent in the TPM and MUST not change upon execution of TPM_Startup(any) command.

Actions

1. If disable is TRUE the following commands will execute with their normal protections
   a. The Avail Disabled column in the ordinal table indicates which commands can and cannot execute
   b. If the command is not available the TPM MUST return TPM_DISABLED upon any attempt to execute the ordinal
   c. TSC_PhysicalPresence can execute when the TPM is disabled
   d. A disabled TPM never prevents the extend capabilities from operating. This is necessary in order to ensure that the records of sequences of integrity metrics in a TPM are always up-to-date. It is irrelevant whether an inactive TPM prevents the extend capabilities from operating, because PCR values cannot be used until the platform is rebooted, at which point existing PCR values are discarded
2. If ownership has the value of FALSE, then any attempt to install an owner fails with the error value TPM_INSTALL_DISABLED.
3. If deactivated is TRUE
   a. This flag does not directly cause capabilities to return the error code TPM_DEACTIVATED.
   b. TPM_Startup uses this flag to set the state of TPM_STCLEAR_FLAGS -> deactivated when the TPM is booted in the state stType==TPM_ST_CLEAR. Only TPM_STCLEAR_FLAGS -> deactivated determines whether capabilities will return the error code TPM_DEACTIVATED.
   c. A change in TPM_PERMANENT_FLAGS -> deactivated therefore has no effect on whether capabilities will return the error code TPM_DEACTIVATED until the next execution of TPM_Startup(ST_CLEAR)
4. If readPubek is TRUE then the TPM_ReadPubek will return the PUBEK, if FALSE the command will return TPM_DISABLED_CMD.
5. If disableOwnerClear is TRUE then TPM_OwnerClear will return TPM_CLEAR_DISABLED, if false the commands will execute.
6. The physicalPresenceHWEnable and physicalPresenceCMDEnable flags MUST mask their respective signals before further processing. The hardware signal, if enabled by the physicalPresenceHWEnable flag, MUST be logically ORed with the PhysicalPresence flag, if enabled, to obtain the final physical presence value used to allow or disallow local commands.

9.1.1 Flag Restrictions

Table 49: TPM_PERMANENT_FLAGS restrictions

Flag SubCap number 0x00000000 +	**Set**	**Set restrictions**	**Actions from**
+1 TPM_PF_DISABLE	Y	Owner authorization or physical presence	TPM_OwnerSetDisable TPM_PhysicalEnable TPM_PhysicalDisable
+2 TPM_PF_OWNERSHIP	Y	No authorization. No ownerinstalled. Physical presence asserted Not available when TPM deactivated or disabled	TPM_SetOwnerInstall
+3 TPM_PF_DEACTIVATED	Y	No authorization, physical presence assertion Not available when TPM disabled	TPM_PhysicalSetDeactivated
+4 TPM_PF_READPUBEK	Y	Owner authorization. Not available when TPM deactivated or disabled	
+5 TPM_PF_DISABLEOWNERCLEAR	Y	Owner authorization. Can only set to TRUE. After being set only ForceClear resets back to FALSE. Not available when TPM deactivated or disabled	TPM_DisableOwnerClear
+6 TPM_PF_ALLOWMAINTENANCE	Y	Owner authorization. Can only set to FALSE, TRUE invalid value. After being set only changing TPM owner resets back to TRUE Not available when TPM deactivated or disabled	TPM_KillMaintenanceFeature
+7 TPM_PF_PHYSICALPRESENCELIFETIMELOCK	N		
+8 TPM_PF_PHYSICALPRESENCEHWENABLE	N		
+9 TPM_PF_PHYSICALPRESENCECMDENABLE	N		
+10 TPM_PF_CEKPUSED	N		
+11 TPM_PF_TPMPOST	N		
+12 TPM_PF_TPMPOSTLOCK	N		
+13 TPM_PF_FIPS	N		
+14 TPM_PF_OPERATOR	N		
+15 TPM_PF_ENABLEREVOKEEK	N		
+16 TPM_PF_NV_LOCKED	N		
+17 TPM_PF_READSRKPUB	Y	Owner Authorization Not available when TPM deactivated or disabled	TPM_SetCapability
+18 TPM_PF_TPMESTABLISHED	Y	Locality 3 or locality 4. Can only set to FALSE.	TSC_ResetEstablishmentBit
+19 TPM_PF_MAINTENANCEDONE	N		
+20 TPM_PF_DISABLEFULLDALOGICINFO	Y	Owner Authorization	TPM_SetCapability

9.2 TPM_STCLEAR_FLAGS

Start of Informative comment

These flags maintain state that is reset on each TPM_Startup(ST_CLEAR) command. The values are not affected by TPM_Startup(ST_STATE) commands.

End of informative comment

```
#define TPM_MAX_FAMILY 8

typedef struct tdTPM_STCLEAR_FLAGS{
   TPM_STRUCTURE_TAG tag;
   BOOL deactivated;
   BOOL disableForceClear;
   BOOL physicalPresence;
   BOOL physicalPresenceLock;
   BOOL bGlobalLock;
} TPM_STCLEAR_FLAGS;
```

Parameters

Table 50: TPM_STCLEAR_FLAGS parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STCLEAR_FLAGS	
BOOL	deactivated	Prevents the operation of most capabilities. There is no default state. It is initialized by TPM_Startup to the same value as TPM_PERMANENT_FLAGS -> deactivated or a set value depending on the type of TPM_Startup. TPM_SetTempDeactivated sets it to TRUE.	TPM_SF_DEACTIVATED
BOOL	disableForceClear	Prevents the operation of TPM_ForceClear when TRUE. The default state is FALSE. TPM_DisableForceClear sets it to TRUE.	TPM_SF_DISABLEFORCECLEAR
BOOL	physicalPresence	Software indication whether an Owner is physically present. The default state is FALSE (Owner is not physically present)	TPM_SF_PHYSICALPRESENCE
BOOL	physicalPresenceLock	Indicates whether changes to the physicalPresence flag are permitted. TPM_Startup/ST_CLEAR sets PhysicalPresence to its default state of FALSE (allow changes to PhysicalPresence flag). The meaning of TRUE is: Do not allow further changes to PhysicalPresence flag. TSC_PhysicalPresence can change the state of physicalPresenceLock.	TPM_SF_PHYSICALPRESENCELOCK
BOOL	bGlobalLock	Set to FALSE on each TPM_Startup(ST_CLEAR). Set to TRUE when a write to NV_Index =0 is successful	TPM_SF_BGLOBALLOCK

Description

1. These values MUST reset upon execution of TPM_Startup(ST_CLEAR).
2. These values MUST NOT reset upon execution of TPM_Startup(ST_STATE).
3. Upon execution of TPM_Startup(ST_DEACTIVATED), all values must be reset except the 'deactivated' flag.

Actions

1. If deactivated is TRUE the following commands SHALL execute with their normal protections
   a. The Avail Deactivated column in the ordinal table indicates which commands can and cannot execute
   b. If the command is not available the TPM MUST return TPM_DEACTIVATED upon any attempt to execute the ordinal
   c. TSC_PhysicalPresence can execute when deactivated
   d. TPM_Extend and TPM_SHA1CompleteExtend MAY execute with their normal protections
2. If disableForceClear is TRUE then the TPM_ForceClear command returns TPM_CLEAR_DISABLED, if FALSE then the command will execute.
3. If physicalPresence is TRUE and TPM_PERMANENT_FLAGS -> physicalPresenceCMDEnable is TRUE, the TPM MAY assume that the Owner is physically present.
4. If physicalPresenceLock is TRUE, TSC_PhysicalPresence MUST NOT change the physicalPresence flag. If physicalPresenceLock is FALSE, TSC_PhysicalPresence will operate.
   a. Set physicalPresenceLock to TRUE at TPM manufacture.

9.2.1 Flag Restrictions

Table 51: TPM_STCLEAR_FLAGS restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_SF_DEACTIVATED	N		
+2 TPM_SF_DISABLEFORCECLEAR	Y	Not available when TPM deactivated or disabled. Can only set to TRUE.	TPM_DisableForceClear
+3 TPM_SF_PHYSICALPRESENCE	N		
+4 TPM_SF_PHYSICALPRESENCELOCK	N		
+5 TPM_SF_BGLOBALLOCK	N		

9.3 TPM_STANY_FLAGS

Start of Informative comment

These flags reset on any TPM_Startup command.

postInitialise indicates only that TPM_Startup has run, not that it was successful.

TOSPresent indicates the presence of a Trusted Operating System (TOS) that was established using the TPM_HASH_START command in the TPM Interface.

End of informative comment

```
typedef struct tdTPM_STANY_FLAGS{
   TPM_STRUCTURE_TAG tag;
   BOOL postInitialise;
   TPM_MODIFIER_INDICATOR localityModifier;
   BOOL transportExclusive;
   BOOL TOSPresent;
} TPM_STANY_FLAGS;
```

Parameters

Table 52: TPM_STANY_FLAGS parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STANY_FLAGS	
BOOL	postInitialise	Prevents the operation of most capabilities. There is no default state. It is initialized by TPM_Init to TRUE. TPM_Startup sets it to FALSE.	TPM_AF_POSTINITIALISE
TPM_MODIFIER_INDICATOR	localityModifier	This SHALL indicate for each command the presence of a locality modifier for the command. It MUST be always ensured that the value during usage reflects the currently active locality.	TPM_AF_LOCALITYMODIFIER
BOOL	transportExclusive	Defaults to FALSE. TRUE when there is an exclusive transport session active. Execution of ANY command other than TPM_ExecuteTransport targeting the exclusive transport session MUST invalidate the exclusive transport session.	TPM_AF_TRANSPORTEXCLUSIVE
BOOL	TOSPresent	Defaults to FALSE Set to TRUE on TPM_HASH_START set to FALSE using setCapability	TPM_AF_TOSPRESENT

Description

This structure MUST reset on TPM_Startup(any)

Actions

1. If postInitialise is TRUE, TPM_Startup SHALL execute as normal
   a. All other commands SHALL return TPM_INVALID_POSTINIT
2. localityModifier is set upon receipt of each command to the TPM. The localityModifier MUST be cleared when the command execution response is read
3. If transportExclusive is TRUE
4. If a command invalidates the exclusive transport session, the command MUST still execute.

5. If TPM_EstablishTransport specifies an exclusive transport session, the existing session is invalidated, a new session is created, and transportExclusive remains TRUE.

9.3.1 Flag Restrictions

Table 53: TPM_STANY_FLAGS restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_AF_POSTINITIALISE	N		
+2 TPM_AF_LOCALITYMODIFIER	N		
+3 TPM_AF_TRANSPORTEXCLUSIVE	N		
+4 TPM_AF_TOSPRESENT	Y	Locality 3 or 4, can only set to FALSE Not available when TPM deactivated or disabled	TPM_SetCapability

9.4 TPM_PERMANENT_DATA

Start of Informative comment

This is an informative structure and not normative. It is purely for convenience of writing the spec.

This structure contains the data fields that are permanently held in the TPM and not affected by TPM_Startup(any).

Many of these fields contain highly confidential and privacy sensitive material. The TPM must maintain the protections around these fields.

End of informative comment

Definition

```
#define TPM_MIN_COUNTERS 4 // the minimum number of counters is 4
#define TPM_DELEGATE_KEY TPM_KEY
#define TPM_NUM_PCR 16
#define TPM_MAX_NV_WRITE_NOOWNER 64

typedef struct tdTPM_PERMANENT_DATA{
   TPM_STRUCTURE_TAG        tag;
   BYTE                     revMajor;
   BYTE                     revMinor;
   TPM_PROOF                tpmProof;
   TPM_NONCE                ekReset;
   TPM_SECRET               ownerAuth;
   TPM_SECRET               operatorAuth;
   TPM_DIRVALUE             authDIR[1];
   TPM_PUBKEY               manuMaintPub;
   TPM_KEY                  endorsementKey;
   TPM_KEY                  srk;
   TPM_KEY                  contextKey;
   TPM_KEY                  delegateKey;
   TPM_COUNTER_VALUE        auditMonotonicCounter;
   TPM_COUNTER_VALUE        monotonicCounter[TPM_MIN_COUNTERS];
   TPM_PCR_ATTRIBUTES       pcrAttrib[TPM_NUM_PCR];
   BYTE                     ordinalAuditStatus[];
   BYTE[]                   rngState;
   TPM_FAMILY_TABLE         familyTable;
   TPM_DELEGATE_TABLE       delegateTable;
   UINT32                   maxNVBufSize;
   UINT32                   lastFamilyID;
   UINT32                   noOwnerNVWrite;
   TPM_CMK_DELEGATE         restrictDelegate;
   TPM_DAA_TPM_SEED         tpmDAASeed;
   TPM_NONCE                daaProof;
   TPM_KEY                  daaBlobKey;
}TPM_PERMANENT_DATA;
```

Parameters

Table 54: TPM_PERMANENT_DATA parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_PERMANENT_DATA	
BYTE	revMajor	This is the TPM major revision indicator. This SHALL be set by the TPME, only. The default value is manufacturer-specific.	TPM_PD_REVMAJOR
BYTE	revMinor	This is the TPM minor revision indicator. This SHALL be set by the TPME, only. The default value is manufacturer-specific.	TPM_PD_REVMINOR
TPM_PROOF	tpmProof	This is a random number that each TPM maintains to validate blobs in the SEAL and other processes. The default value is manufacturer-specific.	TPM_PD_TPMPROOF
TPM_SECRET	ownerAuth	This is the TPM-Owner's AuthData data. The default value is manufacturer-specific.	TPM_PD_OWNERAUTH
TPM_SECRET	operatorAuth	The value that allows the execution of the SetTempDisabled command	TPM_PD_OPERATORAUTH
TPM_PUBKEY	manuMaintPub	This is the manufacturer's public key to use in the maintenance operations. The default value is manufacturer-specific.	TPM_PD_MANUMAINTPUB
TPM_KEY	endorsementKey	This is the TPM's endorsement key pair.	TPM_PD_ENDORSEMENTKEY
TPM_KEY	srk	This is the TPM's StorageRootKey.	TPM_PD_SRK
TPM_KEY	delegateKey	This key encrypts delegate rows that are stored outside the TPM. The key MAY be symmetric or asymmetric. The key size for the algorithm SHOULD be equivalent to 128-bit AES key. The TPM MAY set this value once or allow for changes to this value. This key MUST NOT be the EK or SRK To save space this key MAY be the same key that performs context blob encryption. If an asymmetric algorithm is in use for this key the public portion of the key MUST never be revealed by the TPM. This value MUST be reset when the TPM Owner changes. The value MUST be invalidated with the actions of TPM_OwnerClear. The value MUST be set on TPM_TakeOwnership. The contextKey and delegateKey MAY be the same value.	TPM_PD_DELEGATEKEY
TPM_KEY	contextKey	This is the key in use to perform context saves. The key may be symmetric or asymmetric. The key size is predicated by the algorithm in use. This value MUST be reset when the TPM Owner changes. This key MUST NOT be a copy of the EK or SRK. The contextKey and delegateKey MAY be the same value.	TPM_PD_CONTEXTKEY
TPM_COUNTER_VALUE	auditMonotonicCounter	This SHALL be the audit monotonic counter for the TPM. This value starts at 0 and increments according to the rules of auditing. The label SHALL be fixed at 4 bytes of 0x00.	TPM_PD_AUDITMONOTONICCOUNTER
TPM_COUNTER_VALUE	monotonicCounter	This SHALL be the monotonic counters for the TPM. The individual counters start and increment according to the rules of monotonic counters.	TPM_PD_MONOTONICCOUNTER
TPM_PCR_ATTRIBUTES	pcrAttrib	The attributes for all of the PCR registers supported by the TPM.	TPM_PD_PCRATTRIB
BYTE	ordinalAuditStatus	Table indicating which ordinals are being audited.	TPM_PD_ORDINALAUDITSTATUS

Type	Name	Description	Flag Name
TPM_DIRVALUE	authDIR	The array of TPM Owner authorized DIR. Points to the same location as the NV index value.	TPM_PD_AUTHDIR
BYTE[]	rngState	State information describing the random number generator.	TPM_PD_RNGSTATE
TPM_FAMILY_TABLE	familyTable	The family table in use for delegations	TPM_PD_FAMILYTABLE
TPM_DELEGATE_TABLE	delegateTable	The delegate table	TPM_DELEGATETABLE
TPM_NONCE	ekReset	Nonce held by TPM to validate TPM_RevokeTrust. This value is set as the next 20 bytes from the TPM RNG when the EK is set using TPM_CreateRevocableEK	TPM_PD_EKRESET
UINT32	maxNVBufSize	The maximum size that can be specified in TPM_NV_DefineSpace. This is NOT related to the amount of current NV storage available. This value would be set by the TPM manufacturer and would take into account all of the variables in the specific TPM implementation. Variables could include TPM input buffer max size, transport session overhead, available memory and other factors. The minimum value of maxNVBufSize MUST be 512 and can be larger.	TPM_PD_MAXNVBUFSIZE
UINT32	lastFamilyID	A value that sets the high water mark for family ID's. Set to 0 during TPM manufacturing and never reset.	TPM_PD_LASTFAMILYID
UINT32	noOwnerNVWrite	The count of NV writes that have occurred when there is no TPM Owner. This value starts at 0 in manufacturing and after each TPM_OwnerClear. If the value exceeds 64 the TPM returns TPM_MAXNVWRITES to any command attempting to manipulate the NV storage. Commands that manipulate the NV store are: TPM_Delegate_Manage TPM_Delegate_LoadOwnerDelegation TPM_NV_DefineSpace TPM_NV_WriteValue	TPM_PD_NOOWNERNVWRITE
TPM_CMK_DELEGATE	restrictDelegate	The settings that allow for the delegation and use on CMK keys. Default value is FALSE (0x00000000)	TPM_PD_RESTRICTDELEGATE
TPM_DAA_TPM_SEED	tpmDAASeed	This SHALL be a random value generated after generation of the EK. tpmDAASeed does not change during TPM Owner changes. If the EK is removed (RevokeTrust) then the TPM MUST invalidate the tpmDAASeed. The owner can force a change in the value through TPM_SetCapability.	(linked to daaProof)
TPM_NONCE	daaProof	This is a random number that each TPM maintains to validate blobs in the DAA processes. The default value is manufacturer-specific. The value is not changed when the owner is changed. It is changed when the EK changes. The owner can force a change in the value through TPM_SetCapability.	TPM_PD_DAAPROOF
TPM_KEY	daaBlobKey	This is the key in use to perform DAA encryption and decryption. The key may be symmetric or asymmetric. The key size is predicated by the algorithm in use. This value MUST be changed when daaProof changes. This key MUST NOT be a copy of the EK or SRK.	(linked to daaProof)

9.4.1 Flag Restrictions

Table 55: Flag Restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_PD_REVMAJOR	N		
+2 TPM_PD_REVMINOR	N		
+3 TPM_PD_TPMPROOF	N		
+4 TPM_PD_OWNERAUTH	N		
+5 TPM_PD_OPERATORAUTH	N		
+6 TPM_PD_MANUMAINTPUB	N		
+7 TPM_PD_ENDORSEMENTKEY	N		
+8 TPM_PD_SRK	N		
+9 TPM_PD_DELEGATEKEY	N		
+10 TPM_PD_CONTEXTKEY	N		
+11 TPM_PD_AUDITMONOTONICCOUNTER	N		
+12 TPM_PD_MONOTONICCOUNTER	N		
+13 TPM_PD_PCRATTRIB	N		
+14 TPM_PD_ORDINALAUDITSTATUS	N		
+15 TPM_PD_AUTHDIR	N		
+16 TPM_PD_RNGSTATE	N		
+17 TPM_PD_FAMILYTABLE	N		
+18 TPM_DELEGATETABLE	N		
+19 TPM_PD_EKRESET	N		
+20 TPM_PD_MAXNVBUFSIZE	N		
+21 TPM_PD_LASTFAMILYID	N		
+22 TPM_PD_NOOWNERNVWRITE	N		
+23 TPM_PD_RESTRICTDELEGATE	Y	Owner authorization. Not available when TPM deactivated or disabled.	TPM_CMK_SetRestrictions
+24 TPM_PD_TPMDAASEED	N		
+25 TPM_PD_DAAPROOF	Y	Owner authorization.	

Description

1. TPM_PD_DAAPROOF This capability has no value. When specified by TPM_SetCapability, a new daaProof and daaBlobKey are generated.

9.5 TPM_STCLEAR_DATA

Start of Informative comment

This is an informative structure and not normative. It is purely for convenience of writing the spec.

Most of the data in this structure resets on TPM_Startup(ST_CLEAR). A TPM may implement rules that provide longer-term persistence for the data. The TPM reflects how it handles the data in various TPM_GetCapability fields including startup effects.

End of informative comment

Definition

```
typedef struct tdTPM_STCLEAR_DATA{
   TPM_STRUCTURE_TAG   tag;
   TPM_NONCE           contextNonceKey;
   TPM_COUNT_ID        countID;
   UINT32              ownerReference;
   BOOL                disableResetLock;
    TPM_PCRVALUE       PCR[TPM_NUM_PCR];
   UINT32              deferredPhysicalPresence;

}TPM_STCLEAR_DATA;
```

Parameters

Table 56: TPM_STCLEAR_DATA parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STCLEAR_DATA	
TPM_NONCE	contextNonceKey	This is the nonce in use to properly identify saved key context blobs This SHALL be set to all zeros on each TPM_Startup (ST_Clear).	TPM_SD_CONTEXTNONCEKEY
TPM_COUNT_ID	countID	This is the handle for the current monotonic counter. This SHALL be set to zero on each TPM_Startup(ST_Clear).	TPM_SD_COUNTID
UINT32	ownerReference	Points to where to obtain the owner secret in OIAP and OSAP commands. This allows a TSS to manage 1.1 applications on a 1.2 TPM where delegation is in operation. Default value is TPM_KH_OWNER.	TPM_SD_OWNERREFERENCE
BOOL	disableResetLock	Disables TPM_ResetLockValue upon authorization failure. The value remains TRUE for the timeout period. Default is FALSE. The value is in the STCLEAR_DATA structure as the implementation of this flag is TPM vendor specific.	TPM_SD_DISABLERESETLOCK
TPM_PCRVALUE	PCR	Platform configuration registers	TPM_SD_PCR
UINT32	deferredPhysicalPresence	The value can save the assertion of physicalPresence. Individual bits indicate to its ordinal that physicalPresence was previously asserted when the software state is such that it can no longer be asserted. Set to zero on each TPM_Startup(ST_Clear).	TPM_SD_DEFERREDPHYSICALPRESENCE

Flag Restrictions

Table 57: Flag Restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_SD_CONTEXTNONCEKEY	N		
+2 TPM_SD_COUNTID	N		
+3 TPM_SD_OWNERREFERENCE	N		
+4 TPM_SD_DISABLERESETLOCK	N		
+5 TPM_SD_PCR	N		
+6 TPM_SD_DEFERREDPHYSICALPRESENCE	Y	Can only set to TRUE if PhysicalPresence is asserted. Can set to FALSE at any time.	TPM_SetCapability

Deferred Physical Presence Bit Map

Start of Informative comment

These bits of deferredPhysicalPresence are used in the Actions of the listed ordinals.

Bits are set and cleared using TPM_SetCapability. When physicalPresence is asserted, individual bits can be set or cleared. When physicalPresence is not asserted, individual bits can only be cleared, not set, but bits already set can remain set. Attempting to set a bit when physical presence is not asserted is an error.

End of informative comment

Description

1. If physical presence is not asserted
    a. If TPM_SetCapability -> setValue has a bit set that is not already set in TPM_STCLEAR_DATA -> deferredPhysicalPresence, return TPM_BAD_PRESENCE.
2. Set TPM_STCLEAR_DATA -> deferredPhysicalPresence to TPM_SetCapability -> setValue.

Table 58: Deferred Physical Presence Bit Map

Bit Position	Name	Ordinals Affected
31-1	Unused	Unused
0	unownedFieldUpgrade	TPM_FieldUpgrade

9.6 TPM_STANY_DATA

Start of Informative comment

This is an informative structure and not normative. It is purely for convenience of writing the spec.

Most of the data in this structure resets on TPM_Startup(ST_STATE). A TPM may implement rules that provide longer-term persistence for the data. The TPM reflects how it handles the data in various TPM_GetCapability fields including startup effects.

End of informative comment

Definition

```
#define TPM_MIN_SESSIONS 3
#define TPM_MIN_SESSION_LIST 16

typedef struct tdTPM_SESSION_DATA{
… // vendor specific
} TPM_SESSION_DATA;

typedef struct tdTPM_STANY_DATA{
   TPM_STRUCTURE_TAG   tag;
   TPM_NONCE           contextNonceSession;
   TPM_DIGEST          auditDigest ;
   TPM_CURRENT_TICKS   currentTicks;
   UINT32              contextCount;
   UINT32              contextList[TPM_MIN_SESSION_LIST];
   TPM_SESSION_DATA    sessions[TPM_MIN_SESSIONS];
}TPM_STANY_DATA;
```

Parameters

Table 59: TPM_STANY_DATA parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STANY_DATA	
TPM_NONCE	contextNonceSession	This is the nonce in use to properly identify saved session context blobs. This MUST be set to all zeros on each TPM_Startup (ST_Clear). The nonce MAY be set to all zeros on TPM_Startup(any).	TPM_AD_CONTEXTNONCESESSION
TPM_DIGEST	auditDigest	This is the extended value that is the audit log. This SHALL be set to all zeros at the start of each audit session.	TPM_AD_AUDITDIGEST
TPM_CURRENT_TICKS	currentTicks	This is the current tick counter. This is reset to 0 according to the rules when the TPM can tick. See See Part 1 'Design Section for Time Stamping' for details.	TPM_AD_CURRENTTICKS
UINT32	contextCount	This is the counter to avoid session context blob replay attacks. This MUST be set to 0 on each TPM_Startup (ST_Clear). The value MAY be set to 0 on TPM_Startup (any).	TPM_AD_CONTEXTCOUNT

Type	Name	Description	Flag Name
UINT32	contextList	This is the list of outstanding session blobs. All elements of this array MUST be set to 0 on each TPM_Startup (ST_Clear). The values MAY be set to 0 on TPM_Startup (any). TPM_MIN_SESSION_LIST MUST be 16 or greater.	TPM_AD_CONTEXTLIST
TPM_SESSION_DATA	sessions	List of current sessions. Sessions can be OSAP, OIAP, DSAP and Transport	TPM_AD_SESSIONS

Descriptions

1. The group of contextNonceSession, contextCount, contextList MUST reset at the same time.
2. The contextList MUST keep track of UINT32 values. There is NO requirement that the actual memory be 32 bits
3. contextList MUST support a minimum of 16 entries, it MAY support more.
4. The TPM MAY restrict the absolute difference between contextList entries
   a. For instance if the TPM enforced distance was 10
      i. Entries 8 and 15 would be valid
      ii. Entries 8 and 28 would be invalid
   b. The minimum distance that the TPM MUST support is 2^16, the TPM MAY support larger distances

9.6.1 Flag Restrictions

Table 60: Flag Restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_AD_CONTEXTNONCESESSION	N		
+2 TPM_AD_AUDITDIGEST	N		
+3 TPM_AD_CURRENTTICKS	N		
+4 TPM_AD_CONTEXTCOUNT	N		
+5 TPM_AD_CONTEXTLIST	N		
+6 TPM_AD_SESSIONS	N		

10. PCR Structures

Start of informative comment

The PCR structures expose the information in PCR register, allow for selection of PCR register or registers in the SEAL operation and define what information is held in the PCR register.

These structures are in use during the wrapping of keys and sealing of blobs.

End of informative comment

10.1 TPM_PCR_SELECTION

Start of informative comment

This structure provides a standard method of specifying a list of PCR registers.

This structure provides a means to select some PCR register from the available ones. It is a variable-sized bit vector where each bit corresponds to a specific PCR register. A bit set to 1 indicates that the respective PCR register is selected.

Design points

1. The user needs to be able to specify the null set of PCR. The mask in pcrSelect indicates if a PCR is active or not. Having the mask be a null value that specifies no selected PCR is valid.

2. The TPM must support a sizeOfSelect that indicates the minimum number of PCR on the platform. For a 1.2 PC TPM with 24 PCR this value would be 3.

3. The TPM may support additional PCR over the platform minimum. When supporting additional PCR the TPM must support a sizeOfSelect that can indicate the use of an individual PCR.

4. The TPM may support sizeOfSelect that reflects PCR use other than the maximum. For instance, a PC TPM that supported 48 PCR would require support for a sizeOfSelect of 6 and a sizeOfSelect of 3 (for the 24 required PCR). The TPM could support sizes of 4 and 5.

5. It is desirable for the TPM to support fixed size structures. Nothing in these rules prevents a TPM from only supporting a known set of sizeOfSelect structures.

Odd bit ordering

To the new reader the ordering of the PCR may seem strange. It is. However, the original TPM vendors all interpreted the 1.0 specification to indicate the ordering as it is. The scheme works and is understandable, so to avoid any backwards compatibility no change to the ordering occurs in 1.2. The TPM vendor’s interpretation of the 1.0 specification is the start to the comment that there are no ambiguities in the specification just context sensitive interpretations.

Ordering Example

Consider a TPM with 18 PCR registers. The bit map would contain 3 bytes and the corresponding PCR to bit map would be the following:

```
     Byte 0
+-+-+-+-+-+-+-+-+
|7|6|5|4|3|2|1|0|
+-+-+-+-+-+-+-+-+

      Byte 1
+-+-+-+-+-+-+-+-+
|F|E|D|C|B|A|9|8|
+-+-+-+-+-+-+-+-+

      Byte 2
+--+--+--+--+--+--+--+--+
|17|16|15|14|13|12|11|10|
+--+--+--+--+--+--+--+--+
```

To select PCR 0, pcrSelect would be 00000001

To select PCR 7, pcrSelect would by 10000000

To select PCR 7 and 0, pcrSelect would be 10000001

End of informative comment

Definition

```
typedef struct tdTPM_PCR_SELECTION {
   UINT16 sizeOfSelect;
   [size_is(sizeOfSelect)] BYTE pcrSelect[];
   } TPM_PCR_SELECTION;
```

Parameters

Table 61: TPM_PCR_SELECTION parameters

Type	Name	Description
UINT16	sizeOfSelect	The size in bytes of the pcrSelect structure
BYTE []	pcrSelect	This SHALL be a bit map that indicates if a PCR is active or not

Description

1. PCR selection occurs modulo 8. The minimum granularity for a PCR selection is 8. The specification of registers MUST occur in banks of 8.
2. pcrSelect is a contiguous bit map that shows which PCR are selected. Each byte represents 8 PCR. Byte 0 indicates PCR 0-7, byte 1 8-15 and so on. For each byte, the individual bits represent a corresponding PCR. Refer to the figures below for the mapping of an individual bit to a PCR within a byte. All pcrSelect bytes follow the same mapping.
   a. If the TPM supported 48 PCR to select PCR 0 and 47, the sizeOfSelect would be 6 and only two bits would be set to a 1. The remaining portion of pcrSelect would be NULL
3. When an individual bit is 1 the indicated PCR is selected. If 0 the PCR is not selected.
4. If TPM_PCR_SELECTION.pcrSelect is all 0's
   a. The process MUST set TPM_COMPOSITE_HASH to be all 0's.
5. Else
   a. 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.
6. The TPM MUST support a sizeOfSelect value that reflects the minimum number of PCR as specified in the platform specific specification
7. The TPM MAY return an error if the sizeOfSelect is a value greater than one that represents the number of PCR on the TPM
8. The TPM MUST return an error if sizeOfSelect is 0

10.2 TPM_PCR_COMPOSITE

Start of informative comment

The composite structure provides the index and value of the PCR register to be used when creating the value that SEALS an entity to the composite.

End of informative comment

Definition

```
typedef struct tdTPM_PCR_COMPOSITE {
   TPM_PCR_SELECTION select;
   UINT32 valueSize;
   [size_is(valueSize)] TPM_PCRVALUE pcrValue[];
} TPM_PCR_COMPOSITE;
```

Parameters

Table 62: TPM_PCR_COMPOSITE parameters

Type	Name	Description
TPM_PCR_SELECTION	select	This SHALL be the indication of which PCR values are active
UINT32	valueSize	This SHALL be the size of the pcrValue field (not the number of PCRs)
TPM_PCRVALUE	pcrValue[]	This SHALL be an array of TPM_PCRVALUE structures. The values come in the order specified by the select parameter and are concatenated into a single blob

10.3 TPM_PCR_INFO

Start of informative comment

The TPM_PCR_INFO structure contains the information related to the wrapping of a key or the sealing of data, to a set of PCRs.

End of informative comment

Definition

```
typedef struct tdTPM_PCR_INFO{
   TPM_PCR_SELECTION pcrSelection;
   TPM_COMPOSITE_HASH digestAtRelease;
   TPM_COMPOSITE_HASH digestAtCreation;
} TPM_PCR_INFO;
```

Parameters

Table 63: TPM_PCR_INFO parameters

Type	Name	Description
TPM_PCR_SELECTION	pcrSelection	This SHALL be the selection of PCRs to which the data or key is bound.
TPM_COMPOSITE_HASH	digestAtRelease	This SHALL be the digest of the PCR indices and PCR values to verify when revealing Sealed Data or using a key that was wrapped to PCRs.
TPM_COMPOSITE_HASH	digestAtCreation	This SHALL be the composite digest value of the PCR values, at the time when the sealing is performed.

10.4 TPM_PCR_INFO_LONG

Start of informative comment

The TPM_PCR_INFO structure contains the information related to the wrapping of a key or the sealing of data, to a set of PCRs.

The LONG version includes information necessary to properly define the configuration that creates the blob using the PCR selection.

End of informative comment

Definition

```
typedef struct tdTPM_PCR_INFO_LONG{
   TPM_STRUCTURE_TAG tag;
   TPM_LOCALITY_SELECTION localityAtCreation;
   TPM_LOCALITY_SELECTION localityAtRelease;
   TPM_PCR_SELECTION creationPCRSelection;
   TPM_PCR_SELECTION releasePCRSelection;
   TPM_COMPOSITE_HASH digestAtCreation;
   TPM_COMPOSITE_HASH digestAtRelease;
} TPM_PCR_INFO_LONG;
```

Parameters

Table 64: TPM_PCR_INFO_LONG parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_PCR_INFO_LONG
TPM_LOCALITY_SELECTION	localityAtCreation	This SHALL be the locality modifier when the blob is created
TPM_LOCALITY_SELECTION	localityAtRelease	This SHALL be the locality modifier required to reveal Sealed Data or using a key that was wrapped to PCRs This value MUST not be zero (0).
TPM_PCR_SELECTION	creationPCRSelection	This SHALL be the selection of PCRs active when the blob is created
TPM_PCR_SELECTION	releasePCRSelection	This SHALL be the selection of PCRs to which the data or key is bound.
TPM_COMPOSITE_HASH	digestAtCreation	This SHALL be the composite digest value of the PCR values, when the blob is created
TPM_COMPOSITE_HASH	digestAtRelease	This SHALL be the digest of the PCR indices and PCR values to verify when revealing Sealed Data or using a key that was wrapped to PCRs.

10.5 TPM_PCR_INFO_SHORT

Start of informative comment

This structure is for defining a digest at release when the only information that is necessary is the release configuration.

This structure does not have a tag to keep the structure short. Software and the TPM need to evaluate the structures where the INFO_SHORT structure resides to avoid miss identifying the INFO_SHORT structure.

End of informative comment

Definition

```
typedef struct tdTPM_PCR_INFO_SHORT{
   TPM_PCR_SELECTION pcrSelection;
   TPM_LOCALITY_SELECTION localityAtRelease;
   TPM_COMPOSITE_HASH digestAtRelease;
} TPM_PCR_INFO_SHORT;
```

Parameters

Table 65: TPM_PCR_INFO_SHORT parameters

Type	Name	Description
TPM_PCR_SELECTION	pcrSelection	This SHALL be the selection of PCRs that specifies the digestAtRelease
TPM_LOCALITY_SELECTION	localityAtRelease	This SHALL be the locality modifier required to release the information
TPM_COMPOSITE_HASH	digestAtRelease	This SHALL be the digest of the PCR indices and PCR values to verify when revealing auth data

10.6 TPM_LOCALITY_SELECTION

Start of informative comment

When used with localityAtCreation only one bit is set and it corresponds to the locality of the command creating the structure.

When used with localityAtRelease the bits indicate which localities CAN perform the release.

TPM_LOC_TWO would indicate that only locality 2 can perform the release

TPM_LOC_ONE || TPM_LOC_TWO would indicate that localities 1 or 2 could perform the release

TPM_LOC_FOUR || TPM_LOC_THREE would indicate that localities 3 or 4 could perform the release.

End of informative comment

Definition

```
#define TPM_LOCALITY_SELECTION BYTE
```

Table 66: TPM_LOCALITY_SELECTION definitions

Bit	Name	Description
7:5	Reserved	Must be 0
4	TPM_LOC_FOUR	Locality 4
3	TPM_LOC_THREE	Locality 3
2	TPM_LOC_TWO	Locality 2
1	TPM_LOC_ONE	Locality 1
0	TPM_LOC_ZERO	Locality 0. This is the same as the legacy interface.

The TPM MUST treat a value of 0 as an error. The default value is 0x1F which indicates that localities 0-4 have been selected.

10.7 PCR Attributes

Start of informative comment

Each PCR has attributes associated with it. These attributes allow each PCR to have different behavior. ISO/IEC 11889 defines the generic meaning of the attributes. For a specific platform, the actual setting of the attribute is a platform specific issue.

The attributes are values that are set during the manufacturing process of the TPM and platform and are not field settable or changeable values.

To accommodate debugging, PCR[16] for all platforms will have a certain set of attributes. The setting of these attributes is to allow for easy debugging. This means that values in PCR[16] provide no security information. It is anticipated that PCR[16] would be used by a developer during the development cycle. Developers are responsible for ensuring that a conflict between two programs does not invalidate the settings they are interested in.

The attributes are pcrReset, pcrResetLocal, pcrExtendLocal. Attributes can be set in any combination that is appropriate for the platform.

The pcrReset attribute allows the PCR to be reset at times other than TPM_Startup.

The pcrResetLocal attribute allows the PCR to be reset at times other than TPM_Startup. The reset is legal when the mapping of the command locality to PCR flags results in accept. See 10.8.1 for details.

The pcrExtendLocal attribute modifies the PCR such that the PCR can only be extended when the mapping of the command locality to PCR flags results in accept. See 10.8.1 for details.

End of informative comment

1. The PCR attributes MUST be set during manufacturing.
2. For a specific PCR register, the PCR attributes MUST match the requirements of the TCG platform specific specification that describes the platform.

10.8 TPM_PCR_ATTRIBUTES

Start of informative comment

These attributes are available on a per PCR basis.

The TPM is not required to maintain this structure internally to the TPM.

When a challenger evaluates a PCR, an understanding of this structure is vital to the proper understanding of the platform configuration. As this structure is static for all platforms of the same type, the structure does not need to be reported with each quote.

This normative describes the default response to initialization or a reset. The actual response is platform specific. The platform specification has the final say on the PCR value after initialization or a reset.

End of informative comment

Definition

```
typedef struct tdTPM_PCR_ATTRIBUTES{
   BOOL pcrReset;
   TPM_LOCALITY_SELECTION pcrExtendLocal;
   TPM_LOCALITY_SELECTION pcrResetLocal;
} TPM_PCR_ATTRIBUTES;
```

Types of Persistent Data

Table 67: TPM_PCR_ATTRIBUTES - types of persistent data

Type	Name	Description
BOOL	pcrReset	A value of TRUE SHALL indicate that the PCR register can be reset using the TPM_PCR_Reset command. If pcrReset is: FALSE- Default value of the PCR MUST be 0x00..00 Reset on TPM_Startup(ST_Clear) only Saved by TPM_SaveState Can not be reset by TPM_PCR_Reset TRUE – Default value of the PCR MUST be 0xFF..FF. Reset on TPM_Startup(any) MUST not be part of any state stored by TPM_SaveState Can be reset by TPM_PCR_Reset When reset as part of HASH_START the starting value MUST be 0x00..00
TPM_LOCALITY_SELECTION	pcrResetLocal	An indication of which localities can reset the PCR
TPM_LOCALITY_SELECTION	pcrExtendLocal	An indication of which localities can perform extends on the PCR.

10.8.1 Comparing command locality to PCR flags

Start of informative comment

This is an informative section to show the details of how to check locality against the locality modifier received with a command. The operation works for any of reset, extend or use but for example this will use read.

Map L1 to TPM_STANY_FLAGS -> localityModifier

Map P1 to TPM_PERMANENT_DATA -> pcrAttrib->[selectedPCR].pcrExtendLocal

If, for the value L1, the corresponding bit is set in the bit map P1

return accept

else return reject

End of informative comment

10.9 Debug PCR register

Start of informative comment

There is a need to define a PCR that allows for debugging. The attributes of the debug register are such that it is easy to reset, but the register provides no measurement value that cannot be spoofed. Production applications should not use the debug PCR for any SEAL or other operations. The anticipation is that the debug PCR is set and used by application developers during the application development cycle. Developers are responsible for ensuring that a conflict between two programs does not invalidate the settings they are interested in.

The specific register that is the debug PCR MUST be set by the platform specific specification.

End of informative comment

The attributes for the debug PCR SHALL be the following:

```
pcrReset = TRUE;
pcrResetLocal = 0x1f;
pcrExtendLocal = 0x1f;
pcrUseLocal = 0x1f;
```

These settings are to create a PCR register that developers can use to reset at any time during their development cycle.

The debug PCR does NOT need to be saved during TPM_SaveState

10.10 Mapping PCR Structures

Start of informative comment

When moving information from one PCR structure type to another, i.e. TPM_PCR_INFO to TPM_PCR_INFO_SHORT, the mapping between fields could be ambiguous. This section describes how the various fields map and what the TPM must do when adding or losing information.

End of informative comment

1. Set IN to TPM_PCR_INFO
2. Set IL to TPM_PCR_INFO_LONG
3. Set IS to TPM_PCR_INFO_SHORT
4. To set IS from IN
    a. Set IS -> pcrSelection to IN -> pcrSelection
    b. Set IS -> digestAtRelease to IN -> digestAtRelease
    c. Set IS -> localityAtRelease to 0x1F to indicate all localities are valid
    d. Ignore IN -> digestAtCreation
5. To set IS from IL
    a. Set IS -> pcrSelection to IL -> releasePCRSelection
    b. Set IS -> localityAtRelease to IL -> localityAtRelease
    c. Set IS -> digestAtRelease to IL -> digestAtRelease
    d. Ignore all other IL values
6. To set IL from IN
    a. Set IL -> localityAtCreation to 0x1F
    b. Set IL -> localityAtRelease to 0x1F
    c. Set IL -> creationPCRSelection to IN -> pcrSelection
    d. Set IL -> releasePCRSelection to IN -> pcrSelection
    e. Set IL -> digestAtRelease to IN -> digestAtRelease
    f. Set IL -> digestAtRelease to IN -> digestAtRelease
7. To set IL from IS
    a. Set IL -> localityAtCreation to 0x1F
    b. Set IL -> localityAtRelease to IS localityAtRelease
    c. Set IL -> creationPCRSelection to all zeros
    d. Set IL -> releasePCRSelection to IS -> pcrSelection
    e. Set IL -> digestAtCreation to all zeros
    f. Set IL -> digestAtRelease to IS -> digestAtRelease
8. To set IN from IS
    a. Set IN -> pcrSelection to IS -> pcrSelection
    b. Set IN -> digestAtRelease to IS -> digestAtRelease
    c. Set IN -> digestAtCreation to all zeros

9. To set IN from IL
    a. Set IN -> pcrSelection to IL -> releasePCRSelection
    b. Set IN -> digestAtRelease to IL -> digestAtRelease
    c. If IL -> creationPCRSelection and IL -> localityAtCreation both match IL -> releasePCRSelection and IL –> localityAtRelease
        i. Set IN -> digestAtCreation to IL -> digestAtCreation
    d. Else
        i. Set IN -> digestAtCreation to all zeros

11. Storage Structures

11.1 TPM_STORED_DATA

Start of informative comment

The definition of this structure is necessary to ensure the enforcement of security properties.

This structure is in use by the TPM_Seal and TPM_Unseal commands to identify the PCR index and values that must be present to properly unseal the data.

This structure only provides 1.1 data store and uses TPM_PCR_INFO

End of informative comment

Definition

```
typedef struct tdTPM_STORED_DATA {
   TPM_STRUCT_VER ver;
   UINT32 sealInfoSize;
   [size_is(sealInfoSize)] BYTE[] sealInfo;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE[] encData;
} TPM_STORED_DATA;
```

Parameters

Table 68: TPM_STORED_DATA parameters

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
UINT32	sealInfoSize	Size of the sealInfo parameter
BYTE[]	sealInfo	This SHALL be a structure of type TPM_PCR_INFO or a 0 length array if the data is not bound to PCRs.
UINT32	encDataSize	This SHALL be the size of the encData parameter
BYTE[]	encData	This shall be an encrypted TPM_SEALED_DATA structure containing the confidential part of the data.

Descriptions

1. This structure is created during the TPM_Seal process. The confidential data is encrypted using a non-migratable key. When the TPM_Unseal decrypts this structure the TPM_Unseal uses the public information in the structure to validate the current configuration and release the decrypted data
2. When sealInfoSize is not 0 sealInfo MUST be TPM_PCR_INFO

11.2 TPM_STORED_DATA12

Start of informative comment

The definition of this structure is necessary to ensure the enforcement of security properties.

This structure is in use by the TPM_Seal and TPM_Unseal commands to identify the PCR index and values that must be present to properly unseal the data.

End of informative comment

Definition

```
typedef struct tdTPM_STORED_DATA12 {
   TPM_STRUCTURE_TAG tag;
   TPM_ENTITY_TYPE et;
   UINT32 sealInfoSize;
   [size_is(sealInfoSize)] BYTE[] sealInfo;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE[] encData;
} TPM_STORED_DATA12;
```

Parameters

Table 69: TPM_STORED_DATA12 parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_STORED_DATA12
TPM_ENTITY_TYPE	et	The type of blob
UINT32	sealInfoSize	Size of the sealInfo parameter
BYTE[]	sealInfo	This SHALL be a structure of type TPM_PCR_INFO_LONG
UINT32	encDataSize	This SHALL be the size of the encData parameter
BYTE[]	encData	This shall be an encrypted TPM_SEALED_DATA structure containing the confidential part of the data.

Descriptions

1. This structure is created during the TPM_Seal process. The confidential data is encrypted using a non-migratable key. When the TPM_Unseal decrypts this structure the TPM_Unseal uses the public information in the structure to validate the current configuration and release the decrypted data.
2. If sealInfoSize is not 0 then sealInfo MUST be TPM_PCR_INFO_LONG

11.3 TPM_SEALED_DATA

Start of informative comment

This structure contains confidential information related to sealed data, including the data itself.

End of informative comment

Definition

```
typedef struct tdTPM_SEALED_DATA {
   TPM_PAYLOAD_TYPE payload;
   TPM_SECRET authData;
   TPM_PROOF tpmProof;
   TPM_DIGEST storedDigest;
   UINT32 dataSize;
   [size_is(dataSize)] BYTE[] data;
} TPM_SEALED_DATA;
```

Parameters

Table 70: TPM_SEALED_DATA parameters

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL indicate the payload type of TPM_PT_SEAL
TPM_SECRET	authData	This SHALL be the AuthData data for this value
TPM_PROOF	tpmProof	This SHALL be a copy of TPM_PERMANENT_DATA -> tpmProof
TPM_DIGEST	storedDigest	This SHALL be a digest of the TPM_STORED_DATA structure, excluding the fields TPM_STORED_DATA -> encDataSize and TPM_STORED_DATA -> encData.
UINT32	dataSize	This SHALL be the size of the data parameter
BYTE[]	data	This SHALL be the data to be sealed

Description

1. To tie the TPM_STORED_DATA structure to the TPM_SEALED_DATA structure this structure contains a digest of the containing TPM_STORED_DATA structure.
2. The digest calculation does not include the encDataSize and encData parameters.

11.4 TPM_SYMMETRIC_KEY

Start of informative comment

This structure describes a symmetric key, used during the process “Collating a Request for a Trusted Platform Module Identity”.

End of informative comment

Definition

```
typedef struct tdTPM_SYMMETRIC_KEY {
   TPM_ALGORITHM_ID algId;
   TPM_ENC_SCHEME encScheme;
   UINT16 size;
   [size_is(size)] BYTE[] data;
} TPM_SYMMETRIC_KEY;
```

Parameters

Table 71: TPM_SYMMETRIC_KEY parameters

Type	Name	Description
TPM_ALGORITHM_ID	algId	This SHALL be the algorithm identifier of the symmetric key.
TPM_ENC_SCHEME	encScheme	This SHALL fully identify the manner in which the key will be used for encryption operations.
UINT16	size	This SHALL be the size of the data parameter in bytes
BYTE[]	data	This SHALL be the symmetric key data

11.5 TPM_BOUND_DATA

Start of informative comment

This structure is defined because it is used by a TPM_UnBind command in a consistency check.

The intent of ISO/IEC 11889 is to promote “best practice” heuristics for the use of keys: a signing key shouldn’t be used for storage, and so on. These heuristics are used because of the potential threats that arise when the same key is used in different ways. The heuristics minimize the number of ways in which a given key can be used.

One such heuristic is that a key of type TPM_KEY_BIND, and no other type of key, should always be used to create the blob that is unwrapped by TPM_UnBind. Binding is not a TPM function, so the only choice is to perform a check for the correct payload type when a blob is unwrapped by a key of type TPM_KEY_BIND. This requires the blob to have internal structure.

Even though payloadData has variable size, TPM_BOUND_DATA deliberately does not include the size of payloadData. This is to maximize the size of payloadData that can be encrypted when TPM_BOUND_DATA is encrypted in a single block. When using TPM_UnBind to obtain payloadData, the size of payloadData is deduced as a natural result of the (RSA) decryption process.

End of informative comment

Definition

```
typedef struct tdTPM_BOUND_DATA {
    TPM_STRUCT_VER ver;
    TPM_PAYLOAD_TYPE payload;
    BYTE[] payloadData;
    } TPM_BOUND_DATA;
```

Parameters

Table 72: TPM_BOUND_DATA parameters

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
TPM_PAYLOAD_TYPE	payload	This SHALL be the value TPM_PT_BIND
BYTE[]	payloadData	The bound data

Descriptions

1. This structure MUST be used for creating data when (wrapping with a key of type TPM_KEY_BIND) or (wrapping using the encryption algorithm TPM_ES_RSAESOAEP_SHA1_MGF1). If it is not, the TPM_UnBind command will fail.

12. TPM_KEY complex

Start of informative comment

The TPA_KEY complex is where all of the information regarding keys is kept. These structures combine to fully define and protect the information regarding an asymmetric key.

This version of the specification only fully defines RSA keys, however the design is such that in the future when other asymmetric algorithms are available the general structure will not change.

One overriding design goal is for a 2048 bit RSA key to be able to properly protect another 2048 bit RSA key. This stems from the fact that the SRK is a 2048 bit key and all identities are 2048 bit keys. A goal is to have these keys only require one decryption when loading an identity into the TPM. The structures as defined meet this goal.

Every TPM_KEY is allowed only one encryption scheme or one signature scheme (or one of each in the case of legacy keys) throughout its lifetime. Note however that more than one scheme could be used with externally generated keys, by introducing the same key in multiple blobs.

End of informative comment

12.1 TPM_KEY_PARMS

Start of informative comment

This provides a standard mechanism to define the parameters used to generate a key pair, and to store the parts of a key shared between the public and private key parts.

End of informative comment

Definition

```
typedef struct tdTPM_KEY_PARMS {
   TPM_ALGORITHM_ID algorithmID;
   TPM_ENC_SCHEME encScheme;
   TPM_SIG_SCHEME sigScheme;
   UINT32 parmSize;
   [size_is(parmSize)] BYTE[] parms;
} TPM_KEY_PARMS;
```

Parameters

Table 73: TPM_KEY_PARMS parameters

Type	Name	Description
TPM_ALGORITHM_ID	algorithmID	This SHALL be the key algorithm in use
TPM_ENC_SCHEME	encScheme	This SHALL be the encryption scheme that the key uses to encrypt information
TPM_SIG_SCHEME	sigScheme	This SHALL be the signature scheme that the key uses to perform digital signatures
UINT32	parmSize	This SHALL be the size of the parms field in bytes
BYTE[]	parms	This SHALL be the parameter information dependant upon the key algorithm.

Descriptions

The contents of the ‘parms’ field will vary depending upon algorithmId:

Table 74: TPM_KEY_PARMS descriptions

Algorithm Id	PARMS Contents
TPM_ALG_RSA	A structure of type TPM_RSA_KEY_PARMS
TPM_ALG_SHA	No content
TPM_ALG_HMAC	No content
TPM_ALG_AES	A structure of type TPM_SYMMETRIC_KEY_PARMS
TPM_ALG_MGF1	No content

12.1.1 TPM_RSA_KEY_PARMS

Start of informative comment

This structure describes the parameters of an RSA key.

End of informative comment

Definition

```
typedef struct tdTPM_RSA_KEY_PARMS {
   UINT32 keyLength;
   UINT32 numPrimes;
   UINT32 exponentSize;
   BYTE[] exponent;
} TPM_RSA_KEY_PARMS;
```

Parameters

Table 75: TPM_RSA_KEY_PARMS parameters

Type	Name	Description
UINT32	keyLength	This specifies the size of the RSA key in bits
UINT32	numPrimes	This specifies the number of prime factors used by this RSA key.
UINT32	exponentSize	This SHALL be the size of the exponent. If the key is using the default exponent then the exponentSize MUST be 0.
BYTE[]	exponent	The public exponent of this key

12.1.2 TPM_SYMMETRIC_KEY_PARMS

Start of informative comment

This structure describes the parameters for symmetric algorithms

End of informative comment

Definition

```
typedef struct tdTPM_SYMMETRIC_KEY_PARMS {
   UINT32 keyLength;
   UINT32 blockSize;
   UINT32 ivSize;
    [size_is(ivSize)] BYTE[] IV;
} TPM_SYMMETRIC_KEY_PARMS;
```

Parameters

Table 76: TPM_SYMMETRIC_KEY_PARMS parameters

Type	Name	Description
UINT32	keyLength	This SHALL indicate the length of the key in bits
UINT32	blockSize	This SHALL indicate the block size of the algorithm
UINT32	ivSize	This SHALL indicate the size of the IV
BYTE[]	IV	The initialization vector

12.2 TPM_KEY

Start of informative comment

The TPM_KEY structure provides a mechanism to transport the entire asymmetric key pair. The private portion of the key is always encrypted.

The reason for using a size and pointer for the PCR info structure is save space when the key is not bound to a PCR. The only time the information for the PCR is kept with the key is when the key needs PCR info.

The 1.2 version has a change in the PCRInfo area. For 1.2 the structure uses the TPM_PCR_INFO_LONG structure to properly define the PCR registers in use.

End of informative comment:

Definition

```
typedef struct tdTPM_KEY{
   TPM_STRUCT_VER ver;
   TPM_KEY_USAGE keyUsage;
   TPM_KEY_FLAGS keyFlags;
   TPM_AUTH_DATA_USAGE authDataUsage;
   TPM_KEY_PARMS algorithmParms;
   UINT32 PCRInfoSize;
   [size_is(PCRInfoSize)] BYTE[] PCRInfo;
   TPM_STORE_PUBKEY   pubKey;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE[] encData;
} TPM_KEY;
```

Parameters

Table 77: TPM_KEY parameters

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
TPM_KEY_USAGE	keyUsage	This SHALL be the TPM key usage that determines the operations permitted with this key
TPM_KEY_FLAGS	keyFlags	This SHALL be the indication of migration, redirection etc.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL Indicate the conditions where it is required that authorization be presented.
TPM_KEY_PARMS	algorithmParms	This SHALL be the information regarding the algorithm for this key
UINT32	PCRInfoSize	This SHALL be the length of the pcrInfo parameter. If the key is not bound to a PCR this value SHOULD be 0.
BYTE[]	PCRInfo	This SHALL be a structure of type TPM_PCR_INFO, or an empty array if the key is not bound to PCRs.
TPM_STORE_PUBKEY	pubKey	This SHALL be the public portion of the key
UINT32	encDataSize	This SHALL be the size of the encData parameter.
BYTE[]	encData	This SHALL be an encrypted TPM_STORE_ASYMKEY structure or TPM_MIGRATE_ASYMKEY structure

Version handling

1. A TPM MUST be able to read and create TPM_KEY structures
2. A TPM MUST not allow a TPM_KEY structure to contain a TPM_PCR_INFO_LONG structure

12.3 TPM_KEY12

Start of informative comment

This provides the same functionality as TPM_KEY but uses the new PCR_INFO_LONG structures and the new structure tagging. In all other aspects this is the same structure.

End of informative comment:

Definition

```
typedef struct tdTPM_KEY12{
   TPM_STRUCTURE_TAG tag;
   UINT16 fill;
   TPM_KEY_USAGE keyUsage;
   TPM_KEY_FLAGS keyFlags;
   TPM_AUTH_DATA_USAGE authDataUsage;
   TPM_KEY_PARMS algorithmParms;
   UINT32 PCRInfoSize;
   [size_is(PCRInfoSize)] BYTE[] PCRInfo;
   TPM_STORE_PUBKEY   pubKey;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE[] encData;
} TPM_KEY12;
```

Parameters

Table 78: TPM_KEY12 parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_KEY12
UINT16	fill	MUST be 0x0000
TPM_KEY_USAGE	keyUsage	This SHALL be the TPM key usage that determines the operations permitted with this key
TPM_KEY_FLAGS	keyFlags	This SHALL be the indication of migration, redirection etc.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL Indicate the conditions where it is required that authorization be presented.
TPM_KEY_PARMS	algorithmParms	This SHALL be the information regarding the algorithm for this key
UINT32	PCRInfoSize	This SHALL be the length of the pcrInfo parameter. If the key is not bound to a PCR this value SHOULD be 0.
BYTE[]	PCRInfo	This SHALL be a structure of type TPM_PCR_INFO_LONG,
TPM_STORE_PUBKEY	pubKey	This SHALL be the public portion of the key
UINT32	encDataSize	This SHALL be the size of the encData parameter.
BYTE[]	encData	This SHALL be an encrypted TPM_STORE_ASYMKEY structure TPM_MIGRATE_ASYMKEY structure

Version handling

1. The TPM MUST be able to read and create TPM_KEY12 structures
2. The TPM MUST not allow a TPM_KEY12 structure to contain a TPM_PCR_INFO structure

12.4 TPM_STORE_PUBKEY

Start of informative comment

This structure can be used in conjunction with a corresponding TPM_KEY_PARMS to construct a public key which can be unambiguously used.

End of informative comment

```
typedef struct tdTPM_STORE_PUBKEY {
   UINT32 keyLength;
   BYTE[] key;
} TPM_STORE_PUBKEY;
```

Parameters

Table 79: TPM_STORE_PUBKEY parameters

Type	Name	Description
UINT32	keyLength	This SHALL be the length of the key field.
BYTE[]	key	This SHALL be a structure interpreted according to the algorithm Id in the corresponding TPM_KEY_PARMS structure.

Descriptions

The contents of the ‘key’ field will vary depending upon the corresponding key algorithm:

Table 80: TPM_STORE_PUBKEY algorithm

Algorithm Id	‘Key’ Contents
TPM_ALG_RSA	The RSA public modulus

12.5 TPM_PUBKEY

Start of informative comment

The TPM_PUBKEY structure contains the public portion of an asymmetric key pair. It contains all the information necessary for its unambiguous usage. It is possible to construct this structure from a TPM_KEY, using the algorithmParms and pubKey fields.

End of informative comment

Definition

```
typedef struct tdTPM_PUBKEY{
   TPM_KEY_PARMS algorithmParms;
   TPM_STORE_PUBKEY pubKey;
} TPM_PUBKEY;
```

Parameters

Table 81: TPM_PUBKEY parameters

Type	Name	Description
TPM_KEY_PARMS	algorithmParms	This SHALL be the information regarding this key
TPM_STORE_PUBKEY	pubKey	This SHALL be the public key information

Descriptions

The pubKey member of this structure shall contain the public key for a specific algorithm.

12.6 TPM_STORE_ASYMKEY

Start of informative comment

The TPM_STORE_ASYMKEY structure provides the area to identify the confidential information related to a key. This will include the private key factors for an asymmetric key.

The structure is designed so that encryption of a TPM_STORE_ASYMKEY structure containing a 2048 bit RSA key can be done in one operation if the encrypting key is 2048 bits.

Using typical RSA notation the structure would include P, and when loading the key include the unencrypted P*Q which would be used to recover the Q value.

To accommodate the future use of multiple prime RSA keys the specification of additional prime factors is an optional capability.

This structure provides the basis of defining the protection of the private key.

Changes in this structure MUST be reflected in the TPM_MIGRATE_ASYMKEY structure (section 12.8).

End of informative comment

Definition

```
typedef struct tdTPM_STORE_ASYMKEY {   // pos     len       total
   TPM_PAYLOAD_TYPE payload;           //  0        1          1
   TPM_SECRET usageAuth;               //  1       20         21
   TPM_SECRET migrationAuth;           // 21       20         41
   TPM_DIGEST   pubDataDigest;         // 41       20         61
   TPM_STORE_PRIVKEY privKey;          // 61      132-151   193-214
} TPM_STORE_ASYMKEY;
```

Parameters

Table 82: TPM_STORE_ASYMKEY parameters

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL set to TPM_PT_ASYM to indicate an asymmetric key. If used in TPM_CMK_ConvertMigration the value SHALL be TPM_PT_MIGRATE_EXTERNAL If used in TPM_CMK_CreateKey the value SHALL be TPM_PT_MIGRATE_RESTRICTED
TPM_SECRET	usageAuth	This SHALL be the AuthData data necessary to authorize the use of this value
TPM_SECRET	migrationAuth	This SHALL be the migration AuthData data for a migratable key, or the TPM secret value tpmProof for a non-migratable key created by the TPM. If the TPM sets this parameter to the value tpmProof, then the TPM_KEY.keyFlags.migratable of the corresponding TPM_KEY structure MUST be set to 0. If this parameter is set to the migration AuthData data for the key in parameter PrivKey, then the TPM_KEY.keyFlags.migratable of the corresponding TPM_KEY structure SHOULD be set to 1.
TPM_DIGEST	pubDataDigest	This SHALL be the digest of the corresponding TPM_KEY structure, excluding the fields TPM_KEY.encSize and TPM_KEY.encData. When TPM_KEY -> pcrInfoSize is 0 then the digest calculation has no input from the pcrInfo field. The pcrInfoSize field MUST always be part of the digest calculation.
TPM_STORE_PRIVKEY	privKey	This SHALL be the private key data. The privKey can be a variable length which allows for differences in the key format. The maximum size of the area would be 151 bytes.

12.7 TPM_STORE_PRIVKEY

Start of informative comment

This structure can be used in conjunction with a corresponding TPM_PUBKEY to construct a private key which can be unambiguously used.

End of informative comment

```
typedef struct tdTPM_STORE_PRIVKEY {
   UINT32 keyLength;
   [size_is(keyLength)] BYTE[] key;
} TPM_STORE_PRIVKEY;
```

Parameters

Table 83: TPM_STORE_PRIVKEY parameters

Type	Name	Description
UINT32	keyLength	This SHALL be the length of the key field.
BYTE[]	Key	This SHALL be a structure interpreted according to the algorithm Id in the corresponding TPM_KEY structure.

Descriptions

All migratable keys MUST be RSA keys with two (2) prime factors.

For non-migratable keys, the size, format and contents of privKey.key MAY be vendor specific and MAY not be the same as that used for migratable keys. The level of cryptographic protection MUST be at least as strong as a migratable key.

Table 84: TPM_STORE_PRIVKEY algorithm

Algorithm Id	key Contents
TPM_ALG_RSA	When the numPrimes defined in the corresponding TPM_RSA_KEY_PARMS field is 2, this shall be one of the prime factors of the key. Upon loading of the key the TPM calculates the other prime factor by dividing the modulus, TPM_RSA_PUBKEY, by this value. The TPM MAY support RSA keys with more than two prime factors. Definition of the storage structure for these keys is left to the TPM Manufacturer.

12.8 TPM_MIGRATE_ASYMKEY

Start of informative comment

The TPM_MIGRATE_ASYMKEY structure provides the area to identify the private key factors of a asymmetric key while the key is migrating between TPMs.

This structure provides the basis of defining the protection of the private key.

End of informative comment

Definition

```
typedef struct tdTPM_MIGRATE_ASYMKEY {              // pos   len    total
   TPM_PAYLOAD_TYPE payload;                        //   0     1       1
   TPM_SECRET usageAuth;                            //   1    20      21
   TPM_DIGEST   pubDataDigest;                      //  21    20      41
   UINT32 partPrivKeyLen;                           //  41     4      45
   [size_is(partPrivKeyLen)] BYTE[] partPrivKey;   //  45 112-127  157-172
} TPM_MIGRATE_ASYMKEY;
```

Parameters

Table 85: TPM_MIGRATE_ASYMKEY parameters

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL set to TPM_PT_MIGRATE or TPM_PT_CMK_MIGRATE to indicate an migrating asymmetric key or TPM_PT_MAINT to indicate a maintenance key.
TPM_SECRET	usageAuth	This SHALL be a copy of the usageAuth from the TPM_STORE_ASYMKEY structure.
TPM_DIGEST	pubDataDigest	This SHALL be a copy of the pubDataDigest from the TPM_STORE_ASYMKEY structure.
UINT32	partPrivKeyLen	This SHALL be the size of the partPrivKey field
BYTE[]	partPrivKey	This SHALL be the k2 area as described in TPM_CreateMigrationBlob

12.9 TPM_KEY_CONTROL

Start of informative comment

Attributes that can control various aspects of key usage and manipulation

End of informative comment

Table 86: TPM_KEY_CONTROL parameters

Bit	Name	Description
31:1	Reserved	Must be 0
0	TPM_KEY_CONTROL_OWNER_EVICT	Owner controls when the key is evicted from the TPM. When set the TPM MUST preserve key the key across all TPM_Init invocations.

13. Signed Structures

13.1 TPM_CERTIFY_INFO Structure

Start of informative comment

When the TPM certifies a key, it must provide a signature with a TPM identity key on information that describes that key. This structure provides the mechanism to do so.

Key usage and keyFlags must have their upper byte set to zero to avoid collisions with the other signature headers.

End of informative comment

Definition

```
typedef struct tdTPM_CERTIFY_INFO{
   TPM_STRUCT_VER version;
   TPM_KEY_USAGE keyUsage;
   TPM_KEY_FLAGS keyFlags;
   TPM_AUTH_DATA_USAGE authDataUsage;
   TPM_KEY_PARMS algorithmParms;
   TPM_DIGEST pubkeyDigest;
   TPM_NONCE data;
   BOOL parentPCRStatus;
   UINT32 PCRInfoSize;
   [size_is(pcrInfoSize)] BYTE[] PCRInfo;
} TPM_CERTIFY_INFO;
```

Parameters

Table 87: TPM_CERTIFY_INFO Structure parameters

Type	Name	Description
TPM_STRUCT_VER	version	This MUST be 1.1.0.0
TPM_KEY_USAGE	keyUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified. The upper byte MUST be zero.
TPM_KEY_FLAGS	keyFlags	This SHALL be set to the same value as the corresponding parameter in the TPM_KEY structure that describes the public key that is being certified. The upper byte MUST be zero.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_KEY_PARMS	algorithmParms	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_DIGEST	pubKeyDigest	This SHALL be a digest of the value TPM_KEY -> pubKey -> key in a TPM_KEY representation of the key to be certified
TPM_NONCE	data	This SHALL be externally provided data.
BOOL	parentPCRStatus	This SHALL indicate if any parent key was wrapped to a PCR
UINT32	PCRInfoSize	This SHALL be the size of the pcrInfo parameter. A value of zero indicates that the key is not wrapped to a PCR
BYTE[]	PCRInfo	This SHALL be the TPM_PCR_INFO structure.

13.2 TPM_CERTIFY_INFO2 Structure

Start of informative comment

When the TPM certifies a key, it must provide a signature with a TPM identity key on information that describes that key. This structure provides the mechanism to do so.

Key usage and keyFlags must have their upper byte set to zero to avoid collisions with the other signature headers.

End of informative comment

Definition

```
typedef struct tdTPM_CERTIFY_INFO2{
   TPM_STRUCTURE_TAG tag;
   BYTE fill;
   TPM_PAYLOAD_TYPE payloadType;
   TPM_KEY_USAGE keyUsage;
   TPM_KEY_FLAGS keyFlags;
   TPM_AUTH_DATA_USAGE authDataUsage;
   TPM_KEY_PARMS algorithmParms;
   TPM_DIGEST pubkeyDigest;
   TPM_NONCE data;
   BOOL parentPCRStatus;
   UINT32 PCRInfoSize;
    [size_is(pcrInfoSize)] BYTE[] PCRInfo;
   UINT32 migrationAuthoritySize ;
    [size_is(migrationAuthoritySize)] BYTE[] migrationAuthority;
} TPM_CERTIFY_INFO2;
```

Parameters

Table 88: TPM_CERTIFY_INFO2 Structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CERTIFY_INFO2
BYTE	fill	MUST be 0x00
TPM_PAYLOAD_TYPE	payloadType	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_KEY_USAGE	keyUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified. The upper byte MUST be zero.
TPM_KEY_FLAGS	keyFlags	This SHALL be set to the same value as the corresponding parameter in the TPM_KEY structure that describes the public key that is being certified. The upper byte MUST be zero.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_KEY_PARMS	algorithmParms	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_DIGEST	pubKeyDigest	This SHALL be a digest of the value TPM_KEY -> pubKey -> key in a TPM_KEY representation of the key to be certified
TPM_NONCE	data	This SHALL be externally provided data.
BOOL	parentPCRStatus	This SHALL indicate if any parent key was wrapped to a PCR
UINT32	PCRInfoSize	This SHALL be the size of the pcrInfo parameter.
BYTE[]	PCRInfo	This SHALL be the TPM_PCR_INFO_SHORT structure.
UINT32	migrationAuthoritySize	This SHALL be the size of migrationAuthority
BYTE[]	migrationAuthority	If the key to be certified has [payload == TPM_PT_MIGRATE_RESTRICTED or payload ==TPM_PT_MIGRATE_EXTERNAL], migrationAuthority is the digest of the TPM_MSA_COMPOSITE and has TYPE == TPM_DIGEST. Otherwise it is NULL.

13.3 TPM_QUOTE_INFO Structure

Start of informative comment

This structure provides the mechanism for the TPM to quote the current values of a list of PCRs.

End of informative comment

Definition

```
typedef struct tdTPM_QUOTE_INFO{
   TPM_STRUCT_VER version;
   BYTE[4] fixed;
   TPM_COMPOSITE_HASH digestValue;
   TPM_NONCE externalData;
} TPM_QUOTE_INFO;
```

Parameters

Table 89: TPM_QUOTE_INFO Structure parameters

Type	Name	Description
TPM_STRUCT_VER	version	This MUST be 1.1.0.0
BYTE[4]	fixed	This SHALL always be the string ‘QUOT’
TPM_COMPOSITE_HASH	digestValue	This SHALL be the result of the composite hash algorithm using the current values of the requested PCR indices.
TPM_NONCE	externalData	160 bits of externally supplied data

13.4 TPM_QUOTE_INFO2 Structure

Start of informative comment

This structure provides the mechanism for the TPM to quote the current values of a list of PCRs.

End of informative comment

Definition

```
typedef struct tdTPM_QUOTE_INFO2{
   TPM_STRUCTURE_TAG tag;
   BYTE[4] fixed;
   TPM_NONCE externalData;
   TPM_PCR_INFO_SHORT infoShort;
} TPM_QUOTE_INFO2;
```

Parameters

Table 90: TPM_QUOTE_INFO2 Structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_QUOTE_INFO2
BYTE[4]	fixed	This SHALL always be the string ‘QUT2’
TPM_NONCE	externalData	160 bits of externally supplied data
TPM_PCR_INFO_SHORT	infoShort	the quoted PCR registers

14. Identity Structures

14.1 TPM_EK_BLOB

Start of informative comment

This structure provides a wrapper to each type of structure that will be in use when the endorsement key is in use.

End of informative comment

Definition

```
typedef struct tdTPM_EK_BLOB{
   TPM_STRUCTURE_TAG  tag;
   TPM_EK_TYPE ekType;
   UINT32 blobSize;
   [size_is(blobSize)] byte[] blob;
} TPM_EK_BLOB;
```

Parameters

Table 91: TPM_EK_BLOB parameters

Type	Name	Description
TPM_STRUCTURE_TAG	Tag	TPM_TAG_EK_BLOB
TPM_EK_TYPE	ekType	This SHALL be set to reflect the type of blob in use
UINT32	blobSize	The size of the blob field
BYTE[]	blob	The blob of information depending on the type

14.2 TPM_EK_BLOB_ACTIVATE

Start of informative comment

This structure contains the symmetric key to encrypt the identity credential.

This structure always is contained in a TPM_EK_BLOB.

End of informative comment

Definition

```
typedef struct tdTPM_EK_BLOB_ACTIVATE{
   TPM_STRUCTURE_TAG  tag;
   TPM_SYMMETRIC_KEY sessionKey;
   TPM_DIGEST idDigest;
   TPM_PCR_INFO_SHORT pcrInfo;
} TPM_EK_BLOB_ACTIVATE;
```

Parameters

Table 92: TPM_EK_BLOB_ACTIVATE parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_EK_BLOB_ACTIVATE
TPM_SYMMETRIC_KEY	sessionKey	This SHALL be the session key used by the CA to encrypt the TPM_IDENTITY_CREDENTIAL
TPM_DIGEST	idDigest	This SHALL be the digest of the TPM_PUBKEY that is being certified by the CA
TPM_PCR_INFO_SHORT	pcrInfo	This SHALL indicate the PCRs and localities

14.3 TPM_EK_BLOB_AUTH

Start of informative comment

This structure contains the symmetric key to encrypt the identity credential.

This structure always is contained in a TPM_EK_BLOB.

End of informative comment

Definition

```
typedef struct tdTPM_EK_BLOB_AUTH{
   TPM_STRUCTURE_TAG  tag;
   TPM_SECRET authValue;
} TPM_EK_BLOB_AUTH;
```

Parameters

Table 93: TPM_EK_BLOB_AUTH parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_EK_BLOB_AUTH
TPM_SECRET	authValue	This SHALL be the AuthData value

14.4 TPM_CHOSENID_HASH

This definition specifies the operation necessary to create a TPM_CHOSENID_HASH structure.

Parameters

Table 94: TPM_CHOSENID_HASH parameters

Type	Name	Description
BYTE [ ]	identityLabel	The label chosen for a new TPM identity
TPM_PUBKEY	privacyCA	The public key of a TTP chosen to attest to a new TPM identity

Action

3. TPM_CHOSENID_HASH = SHA(identityLabel || privacyCA)

14.5 TPM_IDENTITY_CONTENTS

Start of informative comment

TPM_MakeIdentity uses this structure and the signature of this structure goes to a privacy CA during the certification process. There is no reason to update the version as this structure did not change for version 1.2.

End of informative comment

Definition

```
typedef struct tdTPM_IDENTITY_CONTENTS {
   TPM_STRUCT_VER          ver;
   UINT32              ordinal;
   TPM_CHOSENID_HASH   labelPrivCADigest;
   TPM_PUBKEY          identityPubKey;
} TPM_IDENTITY_CONTENTS;
```

Parameters

Table 95: TPM_IDENTITY_CONTENTS parameters

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0. This is the version information for this structure and not the underlying key.
UINT32	ordinal	This SHALL be the ordinal of the TPM_MakeIdentity command.
TPM_CHOSENID_HASH	labelPrivCADigest	This SHALL be the result of hashing the chosen identityLabel and privacyCA for the new TPM identity
TPM_PUBKEY	identityPubKey	This SHALL be the public key structure of the identity key

14.6 TPM_IDENTITY_REQ

Start of informative comment

This structure is sent by the TSS to the Privacy CA to create the identity credential.

This structure is informative only.

End of informative comment

Parameters

Table 96: TPM_IDENTITY_REQ parameters

Type	Name	Description
UINT32	asymSize	This SHALL be the size of the asymmetric encrypted area created by TSS_CollateIdentityRequest
UINT32	symSize	This SHALL be the size of the symmetric encrypted area created by TSS_CollateIdentityRequest
TPM_KEY_PARMS	asymAlgorithm	This SHALL be the parameters for the asymmetric algorithm used to create the asymBlob
TPM_KEY_PARMS	symAlgorithm	This SHALL be the parameters for the symmetric algorithm used to create the symBlob
BYTE*	asymBlob	This SHALL be the asymmetric encrypted area from TSS_CollateIdentityRequest
BYTE*	symBlob	This SHALL be the symmetric encrypted area from TSS_CollateIdentityRequest

14.7 TPM_IDENTITY_PROOF

Start of informative comment

Structure in use during the AIK credential process.

End of informative comment

Table 97: TPM_IDENTITY_PROOF parameters

Type	**Name**	**Description**
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
UINT32	labelSize	This SHALL be the size of the label area
UINT32	identityBindingSize	This SHALL be the size of the identitybinding area
UINT32	endorsementSize	This SHALL be the size of the endorsement credential
UINT32	platformSize	This SHALL be the size of the platform credential
UINT32	conformanceSize	This SHALL be the size of the conformance credential
TPM_PUBKEY	identityKey	This SHALL be the public key of the new identity
BYTE*	labelArea	This SHALL be the text label for the new identity
BYTE*	identityBinding	This SHALL be the signature value of TPM_IDENTITY_CONTENTS structure from the TPM_MakeIdentity command
BYTE*	endorsementCredential	This SHALL be the TPM endorsement credential
BYTE*	platformCredential	This SHALL be the TPM platform credential
BYTE*	conformanceCredential	This SHALL be the TPM conformance credential

14.8 TPM_ASYM_CA_CONTENTS

Start of informative comment

This structure contains the symmetric key to encrypt the identity credential.

End of informative comment

Definition

```
typedef struct tdTPM_ASYM_CA_CONTENTS{
   TPM_SYMMETRIC_KEY sessionKey;
   TPM_DIGEST idDigest;
} TPM_ASYM_CA_CONTENTS;
```

Parameters

Table 98: TPM_ASYM_CA_CONTENTS parameters

Type	Name	Description
TPM_SYMMETRIC_KEY	sessionKey	This SHALL be the session key used by the CA to encrypt the TPM_IDENTITY_CREDENTIAL
TPM_DIGEST	idDigest	This SHALL be the digest of the TPM_PUBKEY of the key that is being certified by the CA

14.9 TPM_SYM_CA_ATTESTATION

Start of informative comment

This structure returned by the Privacy CA with the encrypted identity credential.

End of informative comment

Table 99: TPM_SYM_CA_ATTESTATION parameters

Type	Name	Description
UINT32	credSize	This SHALL be the size of the credential parameter
TPM_KEY_PARMS	algorithm	This SHALL be the indicator and parameters for the symmetric algorithm
BYTE[]	credential	This is the result of encrypting TPM_IDENTITY_CREDENTIAL using the session_key and the algorithm indicated “algorithm”

15. Transport structures

15.1 TPM _TRANSPORT_PUBLIC

Start of informative comment

The public information relative to a transport session

End of informative comment

Definition

```
typedef struct tdTPM_TRANSPORT_PUBLIC{
   TPM_STRUCTURE_TAG  tag;
   TPM_TRANSPORT_ATTRIBUTES transAttributes;
   TPM_ALGORITHM_ID algId;
   TPM_ENC_SCHEME encScheme;
} TPM_TRANSPORT_PUBLIC;
```

Parameters

Table 100: TPM _TRANSPORT_PUBLIC parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_PUBLIC
TPM_TRANSPORT_ATTRIBUTES	transAttributes	The attributes of this session
TPM_ALGORITHM_ID	algId	This SHALL be the algorithm identifier of the symmetric key.
TPM_ENC_SCHEME	encScheme	This SHALL fully identify the manner in which the key will be used for encryption operations.

15.1.1 TPM_TRANSPORT_ATTRIBUTES Definitions

Table 101: TPM_TRANSPORT_ATTRIBUTES Definitions

Name	Value	Description
TPM_TRANSPORT_ENCRYPT	0x00000001	The session will provide encryption using the internal encryption algorithm
TPM_TRANSPORT_LOG	0x00000002	The session will provide a log of all operations that occur in the session
TPM_TRANSPORT_EXCLUSIVE	0X00000004	The transport session is exclusive and any command executed outside the transport session causes the invalidation of the session

15.2 TPM_TRANSPORT_INTERNAL

Start of informative comment

The internal information regarding transport session

End of informative comment

Definition

```
typedef struct tdTPM_TRANSPORT_INTERNAL{
   TPM_STRUCTURE_TAG  tag;
   TPM_AUTHDATA authData;
   TPM_TRANSPORT_PUBLIC transPublic;
   TPM_TRANSHANDLE transHandle;
   TPM_NONCE transNonceEven;
   TPM_DIGEST transDigest;
} TPM_TRANSPORT_INTERNAL;
```

Parameters

Table 102: TPM_TRANSPORT_INTERNAL parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_INTERNAL
TPM_AUTHDATA	authData	The shared secret for this session
TPM_TRANSPORT_PUBLIC	transPublic	The public information of this session
TPM_TRANSHANDLE	transHandle	The handle for this session
TPM_NONCE	transNonceEven	The even nonce for the rolling protocol
TPM_DIGEST	transDigest	The log of transport events

15.3 TPM_TRANSPORT_LOG_IN structure

Start of informative comment

The logging of transport commands occurs in two steps, before execution with the input parameters and after execution with the output parameters.

This structure is in use for input log calculations.

End of informative comment

Definition

```
typedef struct tdTPM_TRANSPORT_LOG_IN{
   TPM_STRUCTURE_TAG  tag;
   TPM_DIGEST parameters;
   TPM_DIGEST pubKeyHash;
} TPM_TRANSPORT_LOG_IN;
```

Parameters

Table 103: TPM_TRANSPORT_LOG_IN structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_LOG_IN
TPM_DIGEST	parameters	The actual parameters contained in the digest are subject to the rules of the command using this structure. To find the exact calculation refer to the actions in the command using this structure.
TPM_DIGEST	pubKeyHash	The hash of any keys in the transport command

15.4 TPM_TRANSPORT_LOG_OUT structure

Start of informative comment

The logging of transport commands occurs in two steps, before execution with the input parameters and after execution with the output parameters.

This structure is in use for output log calculations.

This structure is in use for the INPUT logging during releaseTransport.

End of informative comment

Definition

```
typedef struct tdTPM_TRANSPORT_LOG_OUT{
   TPM_STRUCTURE_TAG tag;
   TPM_CURRENT_TICKS currentTicks;
   TPM_DIGEST parameters;
   TPM_MODIFIER_INDICATOR locality;
} TPM_TRANSPORT_LOG_OUT;
```

Parameters

Table 104: TPM_TRANSPORT_LOG_OUT structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_LOG_OUT
TPM_CURRENT_TICKS	currentTicks	The current tick count. This SHALL be the value of the current TPM tick counter.
TPM_DIGEST	parameters	The actual parameters contained in the digest are subject to the rules of the command using this structure. To find the exact calculation refer to the actions in the command using this structure.
TPM_MODIFIER_INDICATOR	locality	The locality that called TPM_ExecuteTransport

15.5 TPM_TRANSPORT_AUTH structure

Start of informative comment

This structure provides the validation for the encrypted AuthData value.

End of informative comment

Definition

```
typedef struct tdTPM_TRANSPORT_AUTH {
   TPM_STRUCTURE_TAG  tag;
   TPM_AUTHDATA authData;
} TPM_TRANSPORT_AUTH;
```

Parameters

Table 105: TPM_TRANSPORT_AUTH structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_AUTH
TPM_AUTHDATA	authData	The AuthData value

16. Audit Structures

16.1 TPM_AUDIT_EVENT_IN structure

Start of informative comment

This structure provides the auditing of the command upon receipt of the command. It provides the information regarding the input parameters.

End of informative comment

Definition

```
typedef struct tdTPM_AUDIT_EVENT_IN {
   TPM_STRUCTURE_TAG  tag;
   TPM_DIGEST inputParms;
   TPM_COUNTER_VALUE auditCount;
} TPM_AUDIT_EVENT_IN;
```

Parameters

Table 106: TPM_AUDIT_EVENT_IN structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_AUDIT_EVENT_IN
TPM_DIGEST	inputParms	Digest value according to the HMAC digest rules of the "above the line" parameters (i.e. the first HMAC digest calculation). When there are no HMAC rules, the input digest includes all parameters including and after the ordinal.
TPM_COUNTER_VALUE	auditCount	The current value of the audit monotonic counter

16.2 TPM_AUDIT_EVENT_OUT structure

Start of informative comment

This structure reports the results of the command execution. It includes the return code and the output parameters.

End of informative comment

Definition

```
typedef struct tdTPM_AUDIT_EVENT_OUT {
   TPM_STRUCTURE_TAG  tag;
   TPM_DIGEST outputParms;
   TPM_COUNTER_VALUE auditCount;
} TPM_AUDIT_EVENT_OUT;
```

Parameters

Table 107: TPM_AUDIT_EVENT_OUT structure parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_AUDIT_EVENT_OUT
TPM_DIGEST	outputParms	Digest value according to the HMAC digest rules of the "above the line" parameters (i.e. the first HMAC digest calculation). When there are no HMAC rules, the output digest includes the return code, the ordinal, and all parameters after the return code.
TPM_COUNTER_VALUE	auditCount	The current value of the audit monotonic counter

17. Tick Structures

17.1 TPM_CURRENT_TICKS

Start of informative comment

This structure holds the current number of time ticks in the TPM. The value is the number of time ticks from the start of the current session. Session start is a variable function that is platform dependent. Some platforms may have batteries or other power sources and keep the TPM clock session across TPM initialization sessions.

The <tickRate> element of the TPM_CURRENT_TICKS structure provides the number of microseconds per tick.

No external entity may ever set the current number of time ticks held in TPM_CURRENT_TICKS. This value is always reset to 0 when a new clock session starts and increments under control of the TPM.

Maintaining the relationship between the number of ticks counted by the TPM and some real world clock is a task for external software.

End of informative comment

Definition

```
typedef struct tdTPM_CURRENT_TICKS {
   TPM_STRUCTURE_TAG  tag;
   UINT64 currentTicks;
   UINT16  tickRate;
   TPM_NONCE     tickNonce;
}TPM_CURRENT_TICKS;
```

Parameters

Table 108: TPM_CURRENT_TICKS parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_CURRENT_TICKS
UINT64	currentTicks	The number of ticks since the start of this tick session
UINT16	tickRate	The number of microseconds per tick. The maximum resolution of the TPM tick counter is thus 1 microsecond. The minimum resolution SHOULD be 1 millisecond.
TPM_NONCE	tickNonce	The nonce created by the TPM when resetting the currentTicks to 0. This indicates the beginning of a time session. This value MUST be valid before the first use of TPM_CURRENT_TICKS. The value can be set at TPM_Startup or just prior to first use.

18. Return codes

Start of informative comment

The TPM has five types of return code. One indicates successful operation and four indicate failure. TPM_SUCCESS (00000000) indicates successful execution. The failure reports are: TPM defined fatal errors (00000001 to 000003FF), vendor defined fatal errors (00000400 to 000007FF), TPM defined non-fatal errors (00000800 to 00000BFF), and vendor defined non-fatal errors (00000C00 to 00000FFF).

The range of vendor defined non-fatal errors was determined by the TSS-WG, which defined XXXX YCCC with XXXX as OS specific and Y defining the TSS SW stack layer (0: TPM layer)

All failure cases return only a non-authenticated fixed set of information. This is because the failure may have been due to authentication or other factors, and there is no possibility of producing an authenticated response.

Fatal errors also terminate any authorization sessions. This is a result of returning only the error code, as there is no way to return the nonces necessary to maintain an authorization session. Non-fatal errors do not terminate authorization sessions.

End of informative comment

Description

1. When a command fails for ANY reason, the TPM MUST return only the following three items:
    a. tag (2 bytes, fixed at TPM_TAG_RSP_COMMAND)
    b. paramSize (4 bytes, fixed at 10)
    c. returnCode (4 bytes, never TPM_SUCCESS)
2. When a command fails, the TPM MUST return a legal error code. Otherwise, the TPM SHOULD return TPM_SUCCESS. If a TPM returns an error code after executing a command, it SHOULD be the error code specified by the command or another legal error code that is appropriate to the error condition.
3. A fatal failure SHALL cause termination of the associated authorization or transport session. A non-fatal failure SHALL NOT cause termination of the associated authorization or transport session.
4. A fatal failure of a wrapped command SHALL not cause any disruption of a transport session that wrapped the failing command. The exception to this is when the failure causes the TPM itself to go into failure mode (selftest failure, etc.)

The return code MUST use the following base. The return code MAY be ISO/IEC 11889 defined or vendor defined.

Mask Parameters

Table 109: Mask Parameters

Name	Value	Description
TPM_BASE	0x0	The start of TPM return codes
TPM_SUCCESS	TPM_BASE	Successful completion of the operation
TPM_VENDOR_ERROR	TPM_Vendor_Specific32	Mask to indicate that the error code is vendor specific for vendor specific commands.
TPM_NON_FATAL	0x00000800	Mask to indicate that the error code is a non-fatal failure.

TPM-defined fatal error codes

Table 110: TPM-defined fatal error codes

Name	Value	Description
TPM_AUTHFAIL	TPM_BASE + 1	Authentication failed
TPM_BADINDEX	TPM_BASE + 2	The index to a PCR, DIR or other register is incorrect
TPM_BAD_PARAMETER	TPM_BASE + 3	One or more parameter is bad
TPM_AUDITFAILURE	TPM_BASE + 4	An operation completed successfully but the auditing of that operation failed.
TPM_CLEAR_DISABLED	TPM_BASE + 5	The clear disable flag is set and all clear operations now require physical access
TPM_DEACTIVATED	TPM_BASE + 6	The TPM is deactivated
TPM_DISABLED	TPM_BASE + 7	The TPM is disabled
TPM_DISABLED_CMD	TPM_BASE + 8	The target command has been disabled
TPM_FAIL	TPM_BASE + 9	The operation failed
TPM_BAD_ORDINAL	TPM_BASE + 10	The ordinal was unknown or inconsistent
TPM_INSTALL_DISABLED	TPM_BASE + 11	The ability to install an owner is disabled
TPM_INVALID_KEYHANDLE	TPM_BASE + 12	The key handle can not be interpreted
TPM_KEYNOTFOUND	TPM_BASE + 13	The key handle points to an invalid key
TPM_INAPPROPRIATE_ENC	TPM_BASE + 14	Unacceptable encryption scheme
TPM_MIGRATEFAIL	TPM_BASE + 15	Migration authorization failed
TPM_INVALID_PCR_INFO	TPM_BASE + 16	PCR information could not be interpreted
TPM_NOSPACE	TPM_BASE + 17	No room to load key.
TPM_NOSRK	TPM_BASE + 18	There is no SRK set
TPM_NOOWNER	TPM_BASE + 18	Added to indicate that no SRK is the equivlent of having no owner
TPM_NOTSEALED_BLOB	TPM_BASE + 19	An encrypted blob is invalid or was not created by this TPM
TPM_OWNER_SET	TPM_BASE + 20	There is already an Owner
TPM_RESOURCES	TPM_BASE + 21	The TPM has insufficient internal resources to perform the requested action.
TPM_SHORTRANDOM	TPM_BASE + 22	A random string was too short
TPM_SIZE	TPM_BASE + 23	The TPM does not have the space to perform the operation.
TPM_WRONGPCRVAL	TPM_BASE + 24	The named PCR value does not match the current PCR value.
TPM_BAD_PARAM_SIZE	TPM_BASE + 25	The paramSize argument to the command has the incorrect value
TPM_SHA_THREAD	TPM_BASE + 26	There is no existing SHA-1 thread.
TPM_SHA_ERROR	TPM_BASE + 27	The calculation is unable to proceed because the existing SHA-1 thread has already encountered an error.
TPM_FAILEDSELFTEST	TPM_BASE + 28	Self-test has failed and the TPM has shutdown.
TPM_AUTH2FAIL	TPM_BASE + 29	The authorization for the second key in a 2 key function failed authorization
TPM_BADTAG	TPM_BASE + 30	The tag value sent to for a command is invalid
TPM_IOERROR	TPM_BASE + 31	An IO error occurred transmitting information to the TPM
TPM_ENCRYPT_ERROR	TPM_BASE + 32	The encryption process had a problem.
TPM_DECRYPT_ERROR	TPM_BASE + 33	The decryption process did not complete.
TPM_INVALID_AUTHHANDLE	TPM_BASE + 34	An invalid handle was used.
TPM_NO_ENDORSEMENT	TPM_BASE + 35	The TPM does not a EK installed
TPM_INVALID_KEYUSAGE	TPM_BASE + 36	The usage of a key is not allowed

Name	Value	Description
TPM_WRONG_ENTITYTYPE	TPM_BASE + 37	The submitted entity type is not allowed
TPM_INVALID_POSTINIT	TPM_BASE + 38	The command was received in the wrong sequence relative to TPM_Init and a subsequent TPM_Startup
TPM_INAPPROPRIATE_SIG	TPM_BASE + 39	Signed data cannot include additional DER information
TPM_BAD_KEY_PROPERTY	TPM_BASE + 40	The key properties in TPM_KEY_PARMs are not supported by this TPM
TPM_BAD_MIGRATION	TPM_BASE + 41	The migration properties of this key are incorrect.
TPM_BAD_SCHEME	TPM_BASE + 42	The signature or encryption scheme for this key is incorrect or not permitted in this situation.
TPM_BAD_DATASIZE	TPM_BASE + 43	The size of the data (or blob) parameter is bad or inconsistent with the referenced key
TPM_BAD_MODE	TPM_BASE + 44	A mode parameter is bad, such as capArea or subCapArea for TPM_GetCapability, physicalPresence parameter for TPM_PhysicalPresence, or migrationType for TPM_CreateMigrationBlob.
TPM_BAD_PRESENCE	TPM_BASE + 45	Either the physicalPresence or physicalPresenceLock bits have the wrong value
TPM_BAD_VERSION	TPM_BASE + 46	The TPM cannot perform this version of the capability
TPM_NO_WRAP_TRANSPORT	TPM_BASE + 47	The TPM does not allow for wrapped transport sessions
TPM_AUDITFAIL_UNSUCCESSFUL	TPM_BASE + 48	TPM audit construction failed and the underlying command was returning a failure code also
TPM_AUDITFAIL_SUCCESSFUL	TPM_BASE + 49	TPM audit construction failed and the underlying command was returning success
TPM_NOTRESETABLE	TPM_BASE + 50	Attempt to reset a PCR register that does not have the resettable attribute
TPM_NOTLOCAL	TPM_BASE + 51	Attempt to reset a PCR register that requires locality and locality modifier not part of command transport
TPM_BAD_TYPE	TPM_BASE + 52	Make identity blob not properly typed
TPM_INVALID_RESOURCE	TPM_BASE + 53	When saving context identified resource type does not match actual resource
TPM_NOTFIPS	TPM_BASE + 54	The TPM is attempting to execute a command only available when in FIPS mode
TPM_INVALID_FAMILY	TPM_BASE + 55	The command is attempting to use an invalid family ID
TPM_NO_NV_PERMISSION	TPM_BASE + 56	The permission to manipulate the NV storage is not available
TPM_REQUIRES_SIGN	TPM_BASE + 57	The operation requires a signed command
TPM_KEY_NOTSUPPORTED	TPM_BASE + 58	Wrong operation to load an NV key
TPM_AUTH_CONFLICT	TPM_BASE + 59	NV_LoadKey blob requires both owner and blob authorization
TPM_AREA_LOCKED	TPM_BASE + 60	The NV area is locked and not writable
TPM_BAD_LOCALITY	TPM_BASE + 61	The locality is incorrect for the attempted operation
TPM_READ_ONLY	TPM_BASE + 62	The NV area is read only and can't be written to
TPM_PER_NOWRITE	TPM_BASE + 63	There is no protection on the write to the NV area
TPM_FAMILYCOUNT	TPM_BASE + 64	The family count value does not match
TPM_WRITE_LOCKED	TPM_BASE + 65	The NV area has already been written to
TPM_BAD_ATTRIBUTES	TPM_BASE + 66	The NV area attributes conflict
TPM_INVALID_STRUCTURE	TPM_BASE + 67	The structure tag and version are invalid or inconsistent
TPM_KEY_OWNER_CONTROL	TPM_BASE + 68	The key is under control of the TPM Owner and can only be evicted by the TPM Owner.
TPM_BAD_COUNTER	TPM_BASE + 69	The counter handle is incorrect
TPM_NOT_FULLWRITE	TPM_BASE + 70	The write is not a complete write of the area

Name	Value	Description
TPM_CONTEXT_GAP	TPM_BASE + 71	The gap between saved context counts is too large
TPM_MAXNVWRITES	TPM_BASE + 72	The maximum number of NV writes without an owner has been exceeded
TPM_NOOPERATOR	TPM_BASE + 73	No operator AuthData value is set
TPM_RESOURCEMISSING	TPM_BASE + 74	The resource pointed to by context is not loaded
TPM_DELEGATE_LOCK	TPM_BASE + 75	The delegate administration is locked
TPM_DELEGATE_FAMILY	TPM_BASE + 76	Attempt to manage a family other then the delegated family
TPM_DELEGATE_ADMIN	TPM_BASE + 77	Delegation table management not enabled
TPM_TRANSPORT_NOTEXCLUSIVE	TPM_BASE + 78	There was a command executed outside of an exclusive transport session
TPM_OWNER_CONTROL	TPM_BASE + 79	Attempt to context save a owner evict controlled key
TPM_DAA_RESOURCES	TPM_BASE + 80	The DAA command has no resources available to execute the command
TPM_DAA_INPUT_DATA0	TPM_BASE + 81	The consistency check on DAA parameter inputData0 has failed.
TPM_DAA_INPUT_DATA1	TPM_BASE + 82	The consistency check on DAA parameter inputData1 has failed.
TPM_DAA_ISSUER_SETTINGS	TPM_BASE + 83	The consistency check on DAA_issuerSettings has failed.
TPM_DAA_TPM_SETTINGS	TPM_BASE + 84	The consistency check on DAA_tpmSpecific has failed.
TPM_DAA_STAGE	TPM_BASE + 85	The atomic process indicated by the submitted DAA command is not the expected process.
TPM_DAA_ISSUER_VALIDITY	TPM_BASE + 86	The issuer's validity check has detected an inconsistency
TPM_DAA_WRONG_W	TPM_BASE + 87	The consistency check on w has failed.
TPM_BAD_HANDLE	TPM_BASE + 88	The handle is incorrect
TPM_BAD_DELEGATE	TPM_BASE + 89	Delegation is not correct
TPM_BADCONTEXT	TPM_BASE + 90	The context blob is invalid
TPM_TOOMANYCONTEXTS	TPM_BASE + 91	Too many contexts held by the TPM
TPM_MA_TICKET_SIGNATURE	TPM_BASE + 92	Migration authority signature validation failure
TPM_MA_DESTINATION	TPM_BASE + 93	Migration destination not authenticated
TPM_MA_SOURCE	TPM_BASE + 94	Migration source incorrect
TPM_MA_AUTHORITY	TPM_BASE + 95	Incorrect migration authority
TPM_PERMANENTEK	TPM_BASE + 97	Attempt to revoke the EK and the EK is not revocable
TPM_BAD_SIGNATURE	TPM_BASE + 98	Bad signature of CMK ticket
TPM_NOCONTEXTSPACE	TPM_BASE + 99	There is no room in the context list for additional contexts

TPM-defined non-fatal errors

Table 111: TPM-defined non-fatal errors

Name	Value	Description
TPM_RETRY	TPM_BASE + TPM_NON_FATAL	The TPM is too busy to respond to the command immediately, but the command could be resubmitted at a later time The TPM MAY return TPM_RETRY for any command at any time.
TPM_NEEDS_SELFTEST	TPM_BASE + TPM_NON_FATAL + 1	TPM_ContinueSelfTest has not been run.
TPM_DOING_SELFTEST	TPM_BASE + TPM_NON_FATAL + 2	The TPM is currently executing the actions of TPM_ContinueSelfTest because the ordinal required resources that have not been tested.
TPM_DEFEND_LOCK_RUNNING	TPM_BASE + TPM_NON_FATAL + 3	The TPM is defending against dictionary attacks and is in some time-out period.

19. Ordinals

Start of informative comment

The command ordinals provide the index value for each command. The following list contains the index value and other information relative to the ordinal.

TPM commands are divided into three classes: Protected/Unprotected, Non-Connection/Connection related, and TPM/Vendor.

End of informative comment

Ordinals are 32 bit values. The upper byte contains values that serve as flag indicators, the next byte contains values indicating what committee designated the ordinal, and the final two bytes contain the Command Ordinal Index.

```
   3                   2                   1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P|C|V| Reserved|    Purview    |   Command Ordinal Index       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

P is Protected/Unprotected command. When 0 the command is a Protected command, when 1 the command is an Unprotected command.

C is Non-Connection/Connection related command. When 0 this command passes through to either the protected (TPM) or unprotected (TSS) components.

V is TPM/Vendor command. When 0 the command is TPM defined, when 1 the command is vendor defined.

All reserved area bits are set to 0.

The following masks are created to allow for the quick definition of the commands

Table 112: Ordinal masks

Value	Event Name	Comments
0x00000000	TPM_PROTECTED_COMMAND	TPM protected command, specified in main specification
0x80000000	TPM_UNPROTECTED_COMMAND	TSS command, specified in the TSS specification
0x40000000	TPM_CONNECTION_COMMAND	TSC command, protected connection commands are specified in the
0x20000000	TPM_VENDOR_COMMAND	Command that is vendor specific for a given TPM or TSS.

The following Purviews have been defined:

Table 113: Ordinal purviews

Value	Event Name	Comments
0x00	TPM_MAIN	Command is from the main specification
0x01	TPM_PC	Command is specific to the PC
0x02	TPM_PDA	Command is specific to a PDA
0x03	TPM_CELL_PHONE	Command is specific to a cell phone
0x04	TPM_SERVER	Command is specific to servers

Combinations for the main specification would be

Table 114: Ordinal combinations

Value	Event Name
TPM_PROTECTED_COMMAND | TPM_MAIN	TPM_PROTECTED_ORDINAL
TPM_UNPROTECTED_COMMAND | TPM_MAIN	TPM_UNPROTECTED_ORDINAL
TPM_CONNECTION_COMMAND | TPM_MAIN	TPM_CONNECTION_ORDINAL

If a command is tagged from the audit column the default state is that use of that command SHALL be audited. Otherwise, the default state is that use of that command SHALL NOT be audited.

Table 115: Column descriptions

Column	Column Values	Comments and valid column entries
AUTH2	x	Does the command support two authorization entities, normally two keys
AUTH1	x	Does the commands support an single authorization session
RQU	x	Does the command execute without any authorization
Optional	O	Is the command optional
No Owner	x	Is the command executable when no owner is present
PCR Use Enforced	x	Does the command enforce PCR restrictions when executed
Physical presence	P	P = The command MAY require physical presence. See the ordinal actions for details.
Audit	X, N	Is the default for auditing enabled N = Never the ordinal is never audited X = Auditing is enabled by default
Duration	S, M, L	What is the expected duration of the command, S = Short implies no asymmetric cryptography M = Medium implies an asymmetric operation L = Long implies asymmetric key generation

1.2 Changes	N, D, X, C	N = New for 1.2
		X = Deleted in 1.2
		D = Deprecated in 1.2
		C = Changed in 1.2

FIPS changes	x	Ordinal has change to satisfy FIPS 140 requirements
Avail Deactivated	x, A	Ordinal will execute when deactivated A = Authorization means that command will only work if the underlying NV store does not require authorization
Avail Disabled	x, A	Ordinal will execute when disabled A = Authorization means that command will only work if the underlying NV store does not require authorization The TPM MUST return TPM_DISABLED for all commands other than those marked as available

The following table is normative, and is the overriding authority in case of discrepancies in other parts of this specification.

Table 116: Ordinal table

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	Physical Presence	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_ActivateIdentity	122	0x0000007A	X	X						X	X	M				
TPM_ORD_AuthorizeMigrationKey	43	0x0000002B		X							X	S				
TPM_ORD_CertifyKey	50	0x00000032	X	X	X					X		M				
TPM_ORD_CertifyKey2	51	0x00000033	X	X	X					X		M	N			
TPM_ORD_CertifySelfTest	82	0x00000052		X	X					X		M	X			
TPM_ORD_ChangeAuth	12	0x0000000C	X							X		M				
TPM_ORD_ChangeAuthAsymFinish	15	0x0000000F		X	X					X		M	D			
TPM_ORD_ChangeAuthAsymStart	14	0x0000000E		X	X					X		L	D			
TPM_ORD_ChangeAuthOwner	16	0x00000010		X						X	X	S				
TPM_ORD_CMK_ApproveMA	29	0x0000001D		X		O						S	N			
TPM_ORD_CMK_ConvertMigration	36	0x00000024		X		O				X		M	N			
TPM_ORD_CMK_CreateBlob	27	0x0000001B		X		O				X		M	N			
TPM_ORD_CMK_CreateKey	19	0x00000013		X		O				X		L	N	X		
TPM_ORD_CMK_CreateTicket	18	0x00000012		X		O						M	N			
TPM_ORD_CMK_SetRestrictions	28	0x0000001C		X		O						S	N			
TPM_ORD_ContinueSelfTest	83	0x00000053			X			X				L		X	X	X
TPM_ORD_ConvertMigrationBlob	42	0x0000002A		X	X					X	X	M				
TPM_ORD_CreateCounter	220	0x000000DC		X								S	N			
TPM_ORD_CreateEndorsementKeyPair	120	0x00000078			X			X				L				
TPM_ORD_CreateMaintenanceArchive	44	0x0000002C		X		O					X	S				
TPM_ORD_CreateMigrationBlob	40	0x00000028	X	X						X	X	M				
TPM_ORD_CreateRevocableEK	127	0x0000007F			X	O		X				L	N			
TPM_ORD_CreateWrapKey	31	0x0000001F		X						X	X	L		X		
TPM_ORD_DAA_Join	41	0x00000029		X		O						L	N			
TPM_ORD_DAA_Sign	49	0x00000031		X		O						L	N			
TPM_ORD_Delegate_CreateKeyDelegation	212	0x000000D4		X								M	N			
TPM_ORD_Delegate_CreateOwnerDelegation	213	0x000000D5		X								M	N			
TPM_ORD_Delegate_LoadOwnerDelegation	216	0x000000D8		X	X			X				M	N			
TPM_ORD_Delegate_Manage	210	0x000000D2		X	X			X				M	N			
TPM_ORD_Delegate_ReadTable	219	0x000000DB			X			X				S	N			
TPM_ORD_Delegate_UpdateVerification	209	0x000000D1		X								S	N			
TPM_ORD_Delegate_VerifyDelegation	214	0x000000D6			X							M	N			
TPM_ORD_DirRead	26	0x0000001A			X							S	D			

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	Physical Presence	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_DirWriteAuth	25	0x00000019		X								S	D			
TPM_ORD_DisableForceClear	94	0x0000005E			X			X			X	S				
TPM_ORD_DisableOwnerClear	92	0x0000005C		X							X	S				
TPM_ORD_DisablePubekRead	126	0x0000007E		X							X	S				
TPM_ORD_DSAP	17	0x00000011			X							S	N		X	X
TPM_ORD_EstablishTransport	230	0x000000E6		X	X					X		S	N			
TPM_ORD_EvictKey	34	0x00000022			X							S	D			
TPM_ORD_ExecuteTransport	231	0x000000E7		X								?L	N			
TPM_ORD_Extend	20	0x00000014			X			X				S			X	X
TPM_ORD_FieldUpgrade	170	0x000000AA	X	X	X	O		X	P			?				
TPM_ORD_FlushSpecific	186	0x000000BA			X			X				S	N		X	X
TPM_ORD_ForceClear	93	0x0000005D			X			X	P		X	S				
TPM_ORD_GetAuditDigest	133	0x00000085			X	O		X			N	S	N			
TPM_ORD_GetAuditDigestSigned	134	0x00000086		X	X	O					N	M	N			
TPM_ORD_GetAuditEvent	130	0x00000082			X	O					N	S	X			
TPM_ORD_GetAuditEventSigned	131	0x00000083		X	X	O					N	M	X			
TPM_ORD_GetCapability	101	0x00000065			X			X				S	C		X	X
TPM_ORD_GetCapabilityOwner	102	0x00000066		X								S	D			
TPM_ORD_GetCapabilitySigned	100	0x00000064		X	X					X		M	X			
TPM_ORD_GetOrdinalAuditStatus	140	0x0000008C			X						N	S	X			
TPM_ORD_GetPubKey	33	0x00000021		X	X					X		S	C			
TPM_ORD_GetRandom	70	0x00000046			X			X				S				
TPM_ORD_GetTestResult	84	0x00000054			X			X				S			X	X
TPM_ORD_GetTicks	241	0x000000F1			X			X				S	N			
TPM_ORD_IncrementCounter	221	0x000000DD		X								S	N			
TPM_ORD_Init	151	0x00000097			X			X				M			X	X
TPM_ORD_KeyControlOwner	35	0x00000023		X								S	N			
TPM_ORD_KillMaintenanceFeature	46	0x0000002E		X		O					X	S				
TPM_ORD_LoadAuthContext	183	0x000000B7			X	O		X				M	D			
TPM_ORD_LoadContext	185	0x000000B9			X							M	N			
TPM_ORD_LoadKey	32	0x00000020		X	X					X		M	D	X		
TPM_ORD_LoadKey2	65	0x00000041		X	X					X		M	N	X		
TPM_ORD_LoadKeyContext	181	0x000000B5			X	O		X				S	D			
TPM_ORD_LoadMaintenanceArchive	45	0x0000002D		X		O					X	S				
TPM_ORD_LoadManuMaintPub	47	0x0000002F			X	O		X			X	S				
TPM_ORD_MakeIdentity	121	0x00000079	X	X						X	X	L		X		
TPM_ORD_MigrateKey	37	0x00000025		X	X					X		M	N			
TPM_ORD_NV_DefineSpace	204	0x000000CC		X	X			X	P			S	N		A	A

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	Physical Presence	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_NV_ReadValue	207	0x000000CF		X	X			X	P	X		S	N		A	A
TPM_ORD_NV_ReadValueAuth	208	0x000000D0		X					P	X		S	N			
TPM_ORD_NV_WriteValue	205	0x000000CD		X	X			X	P	X		S	N		A	A
TPM_ORD_NV_WriteValueAuth	206	0x000000CE		X					P	X		S	N			
TPM_ORD_OIAP	10	0x0000000A			X			X				S			X	X
TPM_ORD_OSAP	11	0x0000000B			X							S			X	X
TPM_ORD_OwnerClear	91	0x0000005B		X							X	S				
TPM_ORD_OwnerReadInternalPub	129	0x00000081		X								S	C			
TPM_ORD_OwnerReadPubek	125	0x0000007D		X							X	S	D			
TPM_ORD_OwnerSetDisable	110	0x0000006E		X							X	S			X	X
TPM_ORD_PCR_Reset	200	0x000000C8			X			X				S	N		X	X
TPM_ORD_PcrRead	21	0x00000015			X			X				S				
TPM_ORD_PhysicalDisable	112	0x00000070			X			X	P		X	S			X	
TPM_ORD_PhysicalEnable	111	0x0000006F			X			X	P		X	S			X	X
TPM_ORD_PhysicalSetDeactivated	114	0x00000072			X			X	P		X	S			X	
TPM_ORD_Quote	22	0x00000016		X	X					X		M				
TPM_ORD_Quote2	62	0x0000003E		X	X	O				X		M	N			
TPM_ORD_ReadCounter	222	0x000000DE			X			X				S	N			
TPM_ORD_ReadManuMaintPub	48	0x00000030			X	O		X			X	S				
TPM_ORD_ReadPubek	124	0x0000007C			X			X			X	S				
TPM_ORD_ReleaseCounter	223	0x000000DF		X				X				S	N			
TPM_ORD_ReleaseCounterOwner	224	0x000000E0		X								S	N			
TPM_ORD_ReleaseTransportSigned	232	0x000000E8	X	X						X		M	N			
TPM_ORD_Reset	90	0x0000005A			X			X				S	C		X	X
TPM_ORD_ResetLockValue	64	0x00000040		X								S	N			
TPM_ORD_RevokeTrust	128	0x00000080			X	O		X	P			S	N			
TPM_ORD_SaveAuthContext	182	0x000000B6			X	O		X				M	D			
TPM_ORD_SaveContext	184	0x000000B8			X							M	N			
TPM_ORD_SaveKeyContext	180	0x000000B4			X	O		X				M	D			
TPM_ORD_SaveState	152	0x00000098			X			X				M			X	X
TPM_ORD_Seal	23	0x00000017		X						X		M				
TPM_ORD_Sealx	61	0x0000003D		X		O				X		M	N			
TPM_ORD_SelfTestFull	80	0x00000050			X			X				L			X	X
TPM_ORD_SetCapability	63	0x0000003F		X	X			X	P			S	N		X	X
TPM_ORD_SetOperatorAuth	116	0x00000074			X			X	P			S	N			
TPM_ORD_SetOrdinalAuditStatus	141	0x0000008D		X		O					X	S				
TPM_ORD_SetOwnerInstall	113	0x00000071			X			X	P		X	S				
TPM_ORD_SetOwnerPointer	117	0x00000075			X							S	N			

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	Physical Presence	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_SetRedirection	154	0x0000009A		X	X	O			P		X	S				
TPM_ORD_SetTempDeactivated	115	0x00000073		X	X			X	P		X	S				
TPM_ORD_SHA1Complete	162	0x000000A2			X			X				S			X	X
TPM_ORD_SHA1CompleteExtend	163	0x000000A3			X			X				S			X	X
TPM_ORD_SHA1Start	160	0x000000A0			X			X				S			X	X
TPM_ORD_SHA1Update	161	0x000000A1			X			X				S			X	X
TPM_ORD_Sign	60	0x0000003C		X	X					X		M				
TPM_ORD_Startup	153	0x00000099			X			X				S			X	X
TPM_ORD_StirRandom	71	0x00000047			X			X				S				
TPM_ORD_TakeOwnership	13	0x0000000D		X				X			X	L			X	
TPM_ORD_Terminate_Handle	150	0x00000096			X			X				S	D		X	X
TPM_ORD_TickStampBlob	242	0x000000F2		X	X					X		M	N			
TPM_ORD_UnBind	30	0x0000001E		X	X					X		M				
TPM_ORD_Unseal	24	0x00000018	X	X						X		M	C			
UNUSED	38	0x00000026														
UNUSED	39	0x00000027														
UNUSED	66	0x00000042														
UNUSED	67	0x00000043														
UNUSED	68	0x00000044														
UNUSED	69	0x00000045														
UNUSED	72	0x00000048														
UNUSED	73	0x00000049														
UNUSED	74	0x0000004A														
UNUSED	75	0x0000004B														
UNUSED	76	0x0000004C														
UNUSED	77	0x0000004D														
UNUSED	78	0x0000004E														
UNUSED	79	0x0000004F														
UNUSED	81	0x00000051														
UNUSED	85	0x00000055														
UNUSED	86	0x00000056														
UNUSED	87	0x00000057														
UNUSED	88	0x00000058														
UNUSED	89	0x00000059														
UNUSED	95	0x0000005F														
UNUSED	96	0x00000060														
UNUSED	97	0x00000061														
UNUSED	98	0x00000062														

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	Physical Presence	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
UNUSED	99	0x00000063														
UNUSED	103	0x00000067														
UNUSED	104	0x00000068														
UNUSED	105	0x00000069														
UNUSED	106	0x0000006A														
UNUSED	107	0x0000006B														
UNUSED	108	0x0000006C														
UNUSED	109	0x0000006D														
UNUSED	118	0x00000076														
UNUSED	119	0x00000077														
UNUSED	132	0x00000084														
UNUSED	135	0x00000087														
UNUSED	136	0x00000088														
UNUSED	137	0x00000089														
UNUSED	138	0x0000008A														
UNUSED	139	0x0000008B														
UNUSED	142	0x0000008E														
UNUSED	143	0x0000008F														
UNUSED	144	0x00000090														
UNUSED	145	0x00000091														
UNUSED	146	0x00000092														
UNUSED	147	0x00000093														
UNUSED	148	0x00000094														
UNUSED	149	0x00000095														
UNUSED	155	0x0000009B														
UNUSED	156	0x0000009C														
UNUSED	157	0x0000009D														
UNUSED	158	0x0000009E														
UNUSED	159	0x0000009F														
UNUSED	164	0x000000A4														
UNUSED	165	0x000000A5														
UNUSED	166	0x000000A6														
UNUSED	167	0x000000A7														
UNUSED	168	0x000000A8														
UNUSED	169	0x000000A9														
UNUSED	171	0x000000AB														
UNUSED	172	0x000000AC														
UNUSED	173	0x000000AD														

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	Physical Presence	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
UNUSED	174	0x000000AE														
UNUSED	175	0x000000AF														
UNUSED	176	0x000000B0														
UNUSED	177	0x000000B1														
UNUSED	178	0x000000B2														
UNUSED	179	0x000000B3														
UNUSED	187	0x000000BB														
UNUSED	188	0x000000BC														
UNUSED	189	0x000000BD														
UNUSED	190	0x000000BE														
UNUSED	191	0x000000BF														
UNUSED	192	0x000000C0														
UNUSED	193	0x000000C1														
UNUSED	194	0x000000C2														
UNUSED	195	0x000000C3														
UNUSED	196	0x000000C4														
UNUSED	197	0x000000C5														
UNUSED	198	0x000000C6														
UNUSED	199	0x000000C7														
UNUSED	202	0x000000CA														
UNUSED	203	0x000000CB														
UNUSED	211	0x000000D3														
UNUSED	215	0x000000D7														
Unused	217	0x000000D9			x							S				
UNUSED	218	0x000000DA														
UNUSED	225	0x000000E1														
UNUSED	233	0x000000E9														
UNUSED	234	0x000000EA														
UNUSED	235	0x000000EB														
UNUSED	236	0x000000EC														
UNUSED	237	0x000000ED														
UNUSED	238	0x000000EE														
UNUSED	239	0x000000EF														
UNUSED	240	0x000000F0														
UNUSED	201	0x00000C9														

19.1 TSC Ordinals

Start of informative comment

The TSC ordinals are optional in the main specification. They are mandatory in the PC Client specification.

End of informative comment

The connection commands manage the TPM's connection to the TBB.

Table 117: TSC Ordinals

	TPM_PROTECTED_ ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	PCR Use enforced	Audit	Duration	1.2 Changes	FIPS Changes	Avail Deactivated	Avail Disabled
TSC_ORD_PhysicalPresence	10	0x4000000A			X	O		X			S	C		X	X
TSC_ORD_ResetEstablishmentBit	11	0x4000000B			X	O		X			S	N		X	X

20. Context structures

20.1 TPM_CONTEXT_BLOB

Start of informative comment

This is the header for the wrapped context. The blob contains all information necessary to reload the context back into the TPM.

The additional data is used by the TPM manufacturer to save information that will assist in the reloading of the context. This area must not contain any shielded data. For instance, the field could contain some size information that allows the TPM more efficient loads of the context. The additional area could not contain one of the primes for a RSA key.

To ensure integrity of the blob when using symmetric encryption the TPM vendor could use some valid cipher chaining mechanism. To ensure the integrity without depending on correct implementation, the TPM_CONTEXT_BLOB structure uses a HMAC of the entire structure using tpmProof as the secret value.

Since both additionalData and sensitiveData are informative, any or all of additionalData could be moved to sensitiveData.

End of informative comment

Definition

```
typedef struct tdTPM_CONTEXT_BLOB {
   TPM_STRUCTURE_TAG tag;
   TPM_RESOURCE_TYPE resourceType;
   TPM_HANDLE handle;
   BYTE[16] label;
   UINT32 contextCount;
   TPM_DIGEST integrityDigest;
   UINT32 additionalSize;
   [size_is(additionalSize)] BYTE[] additionalData;
   UINT32 sensitiveSize;
   [size_is(sensitiveSize)] BYTE[] sensitiveData;
}TPM_CONTEXT_BLOB;
```

Parameters

Table 118: TPM_CONTEXT_BLOB parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CONTEXTBLOB
TPM_RESOURCE_TYPE	resourceType	The resource type
TPM_HANDLE	handle	Previous handle of the resource
BYTE[16]	label	Label for identification of the blob. Free format area.
UINT32	contextCount	MUST be TPM_STANY_DATA -> contextCount when creating the structure. This value is ignored for context blobs that reference a key.
TPM_DIGEST	integrityDigest	The integrity of the entire blob including the sensitive area. This is a HMAC calculation with the entire structure (including sensitiveData) being the hash and tpmProof is the secret
UINT32	additionalSize	The size of additionalData
BYTE[]	additionalData	Additional information set by the TPM that helps define and reload the context. The information held in this area MUST NOT expose any information held in TPM_Shielded-Locations. This should include any IV for symmetric encryption
UINT32	sensitiveSize	The size of sensitiveData
BYTE[]	sensitiveData	The normal information for the resource that can be exported

20.2 TPM_CONTEXT_SENSITIVE

Start of informative comment

The internal areas that the TPM needs to encrypt and store off the TPM.

This is an informative structure and the TPM can implement in any manner they wish.

End of informative comment

Definition

```
typedef struct tdTPM_CONTEXT_SENSITIVE {
   TPM_STRUCTURE_TAG tag;
   TPM_NONCE contextNonce;
   UINT32 internalSize;
   [size_is(internalSize)] BYTE[] internalData;
}TPM_CONTEXT_SENSITIVE;
```

Parameters

Table 119: TPM_CONTEXT_SENSITIVE parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CONTEXT_SENSITIVE
TPM_NONCE	contextNonce	On context blobs other than keys this MUST be TPM_STANY_DATA -> contextNonceSession For keys the value is TPM_STCLEAR_DATA -> contextNonceKey
UINT32	internalSize	The size of the internalData area
BYTE[]	internalData	The internal data area

21. NV storage structures

21.1 TPM_NV_INDEX

Start of informative comment

The index provides the handle to identify the area of storage. The reserved bits allow for a segregation of the index name space to avoid name collisions.

The ISO/IEC 11889 defines the space where the high order bits (T, P, U) are 0. The other spaces are controlled by the indicated entity.

End of informative comment

The TPM_NV_INDEX is a 32-bit value.

```
   3                   2                   1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|T|P|U|D| resvd |   Purview       |         Index                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

1. All reserved area bits are set to 0
   a. T is the TPM manufacturer reserved bit. 0 indicates ISO/IEC 11889 defined value 1 indicates a TPM manufacturer specific value
   b. P is the platform manufacturer reserved bit. 1 indicates that the index controlled by the platform manufacturer.
   c. U is for the platform user. 1 indicates that the index controlled by the platform user.
   d. D indicates defined. 1 indicates that the index is permanently defined and that any defineSpace operation will fail.
   e. ISO/IEC 11889 reserved areas have T/P/U set to 0
   f. ISO/IEC 11889 reserved areas MAY have D set to 0 or 1
2. Purview is the same value used to indicate the platform specific area. This value is the same purview as in use for command ordinals.
   a. The TPM MUST reject index values that do not match the purview of the TPM. This means that a index value for a PDA is rejected by a TPM designed to work on the PC.

21.1.1 Required TPM_NV_INDEX values

Start of informative comment

The required index values must be found on each TPM regardless of platform. These areas are always present and do not require a TPM_NV_DefineSpace command to allocate.

A platform specific specification may add additional required index values for the platform.

End of informative comment

1. The TPM MUST reserve the space as indicated for the required index values

Required Index values

Table 120: Required TPM_NV_INDEX values

Value	Index Name	Default Size	Attributes
0xFFFFFFFF	TPM_NV_INDEX_LOCK	This value turns on the NV authorization protections. Once executed all NV areas us the protections as defined. This value never resets. Attempting to execute TPM_NV_DefineSpace on this value with non-zero size MAY result in a TPM_BADINDEX response.	None
0x00000000	TPM_NV_INDEX0	This value allows for the setting of the bGlobalLock flag, which is only reset on TPM_Startup(ST_Clear) Attempting to execute TPM_NV_WriteValue with a size other than zero MAY result in the TPM_BADINDEX error code.	None
0x10000001	TPM_NV_INDEX_DIR	Size MUST be 20. This index points to the deprecated DIR command area from 1.1. The TPM MUST map this reserved space to be the area operated on by the 1.1 DIR commands. As the DIR commands are deprecated any additional DIR functionally MUST use the NV commands and not the DIR command. Attempts to execute TPM_NV_DefineSpace with this index MUST result in TPM_BADINDEX	TPM_NV_PER_OWNERWRITE TPM_NV_PER_WRITEALL

21.1.2 Reserved Index values

Start of informative comment

The reserved values are defined to avoid index collisions. These values are not in each and every TPM.

End of informative comment

1. The reserved index values are to avoid index value collisions.
2. These index values require a TPM_NV_DefineSpace to have the area for the index allocated
3. A platform specific specification MAY indicate that reserved values are required.
4. The reserved index values MAY have their D bit set by the TPM vendor to permanently reserve the index in the TPM

Table 121: Reserved Index values

Value	Event Name	Default Size
0x0000Fxxx	TPM_NV_INDEX_TPM	Reserved for TPM use
0x0000F000	TPM_NV_INDEX_EKCert	The Endorsement credential
0x0000F001	TPM_NV_INDEX_TPM_CC	The TPM Conformance credential
0x0000F002	TPM_NV_INDEX_PlatformCert	The platform credential
0x0000F003	TPM_NV_INDEX_Platform_CC	The Platform conformance credential
0x0000F004	TPM_NV_INDEX_TRIAL	To try TPM_NV_DefineSpace without actually allocating NV space
0x000111xx	TPM_NV_INDEX_TSS	Reserved for TSS use
0x000112xx	TPM_NV_INDEX_PC	Reserved for PC Client use
0x000113xx	TPM_NV_INDEX_SERVER	reserved for Server use
0x000114xx	TPM_NV_INDEX_MOBILE	Reserved for mobile use
0x000115xx	TPM_NV_INDEX_PERIPHERAL	Reserved for peripheral use
0x000116xx	TPM_NV_INDEX_GPIO_xx	Reserved for GPIO pins
0x0001xxxx	TPM_NV_INDEX_GROUP_RESV	Reserved for TCG WG's

21.2 TPM_NV_ATTRIBUTES

Start of informative comment

This structure allows the TPM to keep track of the data and permissions to manipulate the area.

A write once per lifetime of the TPM attribute, while attractive, is simply too dangerous (attacker allocates all of the NV area and uses it). The locked attribute adds close to that functionality. This allows the area to be "locked" and only changed when unlocked. The lock bit would be set for all indexes sometime during the initialization of a platform. The use model would be that the platform BIOS would lock the TPM and only allow changes in the BIOS setup routine.

There are no locality bits to allow for a locality to define space. The rationale behind this is that the define space includes the permissions so that would mean any locality could define space. The use model for localities would assume that the platform owner was opting into the use of localities and would define the space necessary to operate when the opt-in was authorized.

The attributes TPM_NV_PER_AUTHREAD and TPM_NV_PER_OWNERREAD cannot both be set to TRUE. Similarly, the attributes TPM_NV_PER_AUTHWRITE and TPM_NV_PER_OWNERWRITE cannot both be set to TRUE.

End of informative comment

Definition

```
typedef struct tdTPM_NV_ATTRIBUTES{
   TPM_STRUCTURE_TAG tag;
   UINT32 attributes;
} TPM_NV_ATTRIBUTES;
```

Parameters

Table 122: TPM_NV_ATTRIBUTES parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_NV_ATTRIBUTES
UINT32	attributes	The attribute area

Attributes values

Table 123: TPM_NV_ATTRIBUTES attribute values

Bit	Name	Description
31	TPM_NV_PER_READ_STCLEAR	The value can be read until locked by a read with a data size of 0. It can only be unlocked by TPM_Startup(ST_Clear) or a successful write. Lock held for each area in bReadSTClear.
30:19	Reserved	
18	TPM_NV_PER_AUTHREAD	The value requires authorization to read
17	TPM_NV_PER_OWNERREAD	The value requires TPM Owner authorization to read.
16	TPM_NV_PER_PPREAD	The value requires physical presence to read
15	TPM_NV_PER_GLOBALLOCK	The value is writable until a write to index 0 is successful. The lock of this attribute is reset by TPM_Startup(ST_CLEAR). Lock held by SF -> bGlobalLock
14	TPM_NV_PER_WRITE_STCLEAR	The value is writable until a write to the specified index with a datasize of 0 is successful. The lock of this attribute is reset by TPM_Startup(ST_CLEAR). Lock held for each area in bWriteSTClear.
13	TPM_NV_PER_WRITEDEFINE	Lock set by writing to the index with a datasize of 0. Lock held for each area in bWriteDefine. This is a persistent lock.
12	TPM_NV_PER_WRITEALL	The value must be written in a single operation
11:3	Reserved for write additions	
2	TPM_NV_PER_AUTHWRITE	The value requires authorization to write
1	TPM_NV_PER_OWNERWRITE	The value requires TPM Owner authorization to write
0	TPM_NV_PER_PPWRITE	The value requires physical presence to write

21.3 TPM_NV_DATA_PUBLIC

Start of informative comment

This structure represents the public description and controls on the NV area.

bReadSTClear and bWriteSTClear are volatile, in that they are set FALSE at TPM_Startup(ST_Clear). bWriteDefine is persistent, in that it remains TRUE through startup.

End of informative comment

Definition

```
typedef struct tdTPM_NV_DATA_PUBLIC {
   TPM_STRUCTURE_TAG tag;
   TPM_NV_INDEX nvIndex;
   TPM_PCR_INFO_SHORT pcrInfoRead;
   TPM_PCR_INFO_SHORT pcrInfoWrite;
   TPM_NV_ATTRIBUTES permission;
   BOOL bReadSTClear;
   BOOL bWriteSTClear;
   BOOL bWriteDefine;
   UINT32 dataSize;
} TPM_NV_DATA_PUBLIC;
```

Parameters

Table 124: TPM_NV_DATA_PUBLIC parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_NV_DATA_PUBLIC
TPM_NV_INDEX	nvIndex	The index of the data area
TPM_PCR_INFO_SHORT	pcrInfoRead	The PCR selection that allows reading of the area
TPM_PCR_INFO_SHORT	pcrInfoWrite	The PCR selection that allows writing of the area
TPM_NV_ATTRIBUTES	permission	The permissions for manipulating the area
BOOL	bReadSTClear	Set to FALSE on each TPM_Startup(ST_Clear) and set to TRUE after a ReadValuexxx with datasize of 0
BOOL	bWriteSTClear	Set to FALSE on each TPM_Startup(ST_CLEAR) and set to TRUE after a WriteValuexxx with a datasize of 0.
BOOL	bWriteDefine	Set to FALSE after TPM_NV_DefineSpace and set to TRUE after a successful WriteValuexxx with a datasize of 0
UINT32	dataSize	The size of the data area in bytes

Actions

1. On read of this structure (through TPM_GetCapability) if pcrInfoRead -> pcrSelect is 0 then pcrInfoRead -> digestAtRelease MUST be 0x00…00
2. On read of this structure (through TPM_GetCapability) if pcrInfoWrite -> pcrSelect is 0 then pcrInfoWrite -> digestAtRelease MUST be 0x00…00

21.4 TPM_NV_DATA_SENSITIVE

Start of informative comment

This is an internal structure that the TPM uses to keep the actual NV data and the controls regarding the area.

This entire section is informative

End of informative comment

Definition

```
typedef struct tdTPM_NV_DATA_SENSITIVE {
   TPM_STRUCTURE_TAG tag;
   TPM_NV_DATA_PUBLIC pubInfo;
   TPM_AUTHDATA authValue;
   [size_is(dataSize)] BYTE[] data;
} TPM_NV_DATA_SENSITIVE;
```

Parameters

Table 125: TPM_NV_DATA_SENSITIVE parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_NV_DATA_SENSITIVE
TPM_NV_DATA_PUBLIC	pubInfo	The public information regarding this area
TPM_AUTHDATA	authValue	The AuthData value to manipulate the value
BYTE[]	data	The data area. This MUST not contain any sensitive information as the TPM does not provide any confidentiality on the data.

21.5 Max NV Size

The value TPM_MAX_NV_SIZE is a value where the minimum value is set by the platform specific specification. The TPM vendor can design a TPM with a size that is larger than the minimum.

21.6 TPM_NV_DATA_AREA

Start of informative comment

TPM_NV_DATA_AREA is an indication of the internal structure the TPM uses to track NV areas. The structure definition is TPM vendor specific and never leaves the TPM. The structure would contain both the TPM_NV_DATA_PUBLIC and TPM_NV_DATA_SENSITIVE areas.

End of informative comment

22. Delegate Structures

22.1 Structures and encryption

Start of informative comment

The TPM is responsible for encrypting various delegation elements when stored off the TPM. When the structures are TPM internal structures and not in use by any other process (i.e. TPM_DELEGATE_SENSITIVE) the structure is merely an informative comment as to the information necessary to make delegation work. The TPM may put additional, or possibly, less information into the structure and still obtain the same result.

Where the structures are in use across TPMs or in use by outside processes (i.e. TPM_DELEGATE_PUBLIC), the structure is normative and the must use the structure without modification.

End of informative comment

1. The TPM MUST provide encryption of sensitive areas held outside of the TPM. The encryption MUST be comparable to AES 128-bit key.

22.2 Delegate Definitions

Informative comment

The delegations are in a 64-bit field. Each bit describes a capability that the TPM Owner or an authorized key user can delegate to a trusted process by setting that bit. Each delegation bit setting is independent of any other delegation bit setting in a row.

If a TPM command is not listed in the following table, then the TPM Owner or the key user cannot delegate that capability to a trusted process. For the TPM commands that are listed in the following table, if the bit associated with a TPM command is set to zero in the row of the table that identifies a trusted process, then that process has not been delegated to use that TPM command.

The minimum granularity for delegation is at the ordinal level. It is not possible to delegate an option of an ordinal. This implies that if the options present a difficulty and there is a need to separate the delegations then there needs to be a split into two separate ordinals.

End of informative comment

```
#define TPM_DEL_OWNER_BITS 0x00000001
#define TPM_DEL_KEY_BITS   0x00000002

typedef struct tdTPM_DELEGATIONS{
   TPM_STRUCTURE_TAG tag;
   UINT32 delegateType;
   UINT32 per1;
   UINT32 per2;
} TPM_DELEGATIONS;
```

Parameters

Table 126: Delegate Definitions parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_DELEGATIONS
UINT32	delegateType	Owner or key
UNIT32	per1	The first block of permissions
UINT32	per2	The second block of permissions

22.2.1 Owner Permission Settings

Informative comment

This section is going to remove any ambiguity as to the order of bits in the permission array

End of informative comment

Per1 bits

Table 127: Owner Permission Settings - Per1 bits

Bit Number	Ordinal	Bit Name
31	Reserved	Reserved MUST be 0
30	TPM_ORD_SetOrdinalAuditStatus	TPM_DELEGATE_SetOrdinalAuditStatus
29	TPM_ORD_DirWriteAuth	TPM_DELEGATE_DirWriteAuth
28	TPM_ORD_CMK_ApproveMA	TPM_DELEGATE_CMK_ApproveMA
27	TPM_ORD_NV_WriteValue	TPM_DELEGATE_NV_WriteValue
26	TPM_ORD_CMK_CreateTicket	TPM_DELEGATE_CMK_CreateTicket
25	TPM_ORD_NV_ReadValue	TPM_DELEGATE_NV_ReadValue
24	TPM_ORD_Delegate_LoadOwnerDelegation	TPM_DELEGATE_Delegate_LoadOwnerDelegation
23	TPM_ORD_DAA_Join	TPM_DELEGATE_DAA_Join
22	TPM_ORD_AuthorizeMigrationKey	TPM_DELEGATE_AuthorizeMigrationKey
21	TPM_ORD_CreateMaintenanceArchive	TPM_DELEGATE_CreateMaintenanceArchive
20	TPM_ORD_LoadMaintenanceArchive	TPM_DELEGATE_LoadMaintenanceArchive
19	TPM_ORD_KillMaintenanceFeature	TPM_DELEGATE_KillMaintenanceFeature
18	TPM_ORD_OwnerReadInternalPub	TPM_DELEGATE_OwnerReadInternalPub
17	TPM_ORD_ResetLockValue	TPM_DELEGATE_ResetLockValue
16	TPM_ORD_OwnerClear	TPM_DELEGATE_OwnerClear
15	TPM_ORD_DisableOwnerClear	TPM_DELEGATE_DisableOwnerClear
14	TPM_ORD_NV_DefineSpace	TPM_DELEGATE_NV_DefineSpace
13	TPM_ORD_OwnerSetDisable	TPM_DELEGATE_OwnerSetDisable
12	TPM_ORD_SetCapability	TPM_DELEGATE_SetCapability
11	TPM_ORD_MakeIdentity	TPM_DELEGATE_MakeIdentity
10	TPM_ORD_ActivateIdentity	TPM_DELEGATE_ActivateIdentity
9	TPM_ORD_OwnerReadPubek	TPM_DELEGATE_OwnerReadPubek
8	TPM_ORD_DisablePubekRead	TPM_DELEGATE_DisablePubekRead
7	TPM_ORD_SetRedirection	TPM_DELEGATE_SetRedirection
6	TPM_ORD_FieldUpgrade	TPM_DELEGATE_FieldUpgrade
5	TPM_ORD_Delegate_UpdateVerification	TPM_DELEGATE_Delegate_UpdateVerification
4	TPM_ORD_CreateCounter	TPM_DELEGATE_CreateCounter
3	TPM_ORD_ReleaseCounterOwner	TPM_DELEGATE_ReleaseCounterOwner
2	TPM_ORD_Delegate_Manage	TPM_DELEGATE_Delegate_Manage
1	TPM_ORD_Delegate_CreateOwnerDelegation	TPM_DELEGATE_Delegate_CreateOwnerDelegation
0	TPM_ORD_DAA_Sign	TPM_DELEGATE_DAA_Sign

Per2 bits

Table 128: Owner Permission Settings - Per2 bits

Bit Number	Ordinal	Bit Name
31:0	Reserved	Reserved MUST be 0

22.2.2 Owner commands not delegated

Start of informative comment

Not all TPM Owner authorized commands can be delegated. The following table lists those commands the reason why the command is not delegated.

End of informative comment

Table 129: Owner commands not delegated

Command	Rationale
TPM_ChangeAuthOwner	Delegating change owner allows the delegatee to control the TPM Owner. This implies that the delegate has more control than the owner. The owner can create the same situation by merely having the process that the owner wishes to control the TPM to perform ChangeOwner with the current owner's permission.
TPM_TakeOwnership	If you don't have an owner how can the current owner delegate the command.
TPM_CMK_SetRestrictions	This command allows the owner to restrict what processes can be delegated the ability to create and manipulate CMK keys
TPM_GetCapabilityOwner	This command is deprecated. Its only purpose is to find out/verify the owner password correctness, Therefore it makes no sense to delegate it.