TPM Main
Part 2 TPM Structures

Specification version 1.2
Revision 62
2 October 2003

Contact: tpmwg@trustedcomputinggroup.org

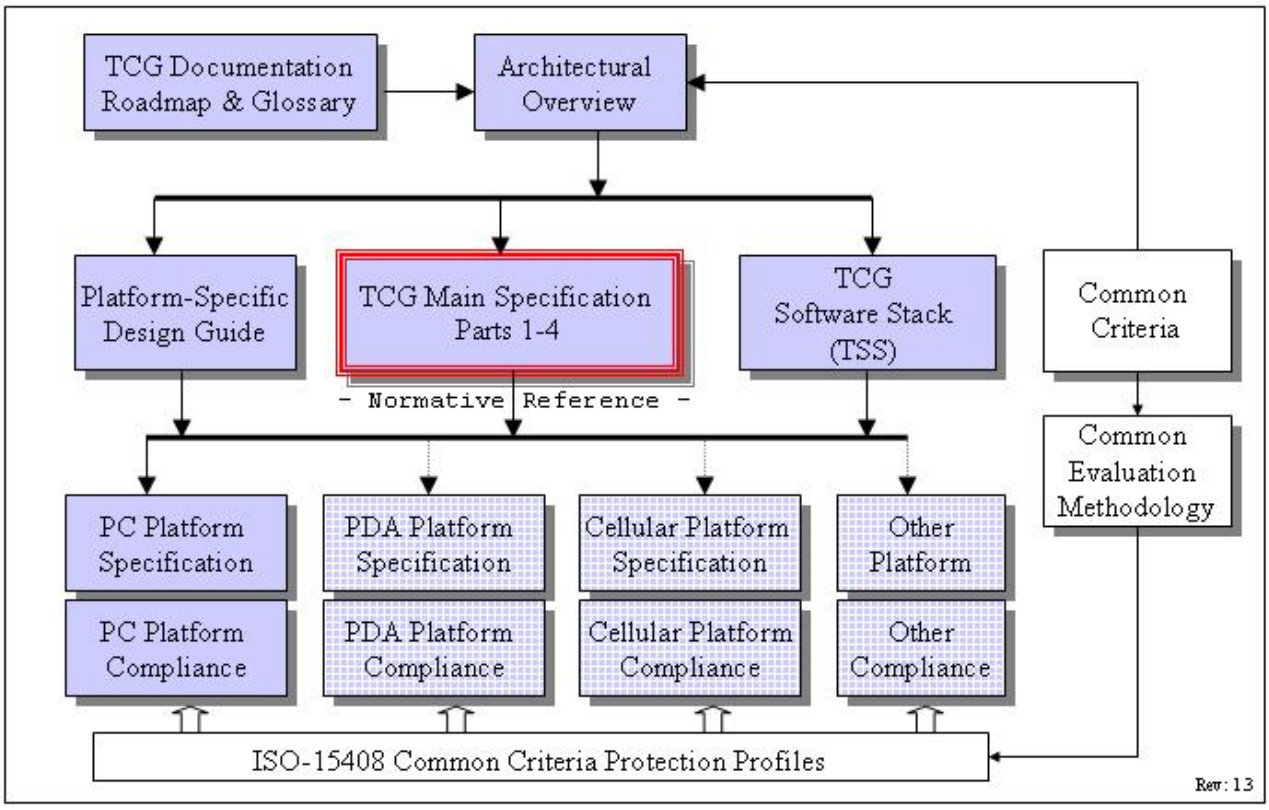
TCG PUBLISHED

TCG

Revision History

.10	Started: 1 April 2003. Last Updated: 04/30/01 by David Grawrock
52	Started 15 July 2003 by David Graworck
53	Started 5 Aug 2003 by David Grawrock

TCG Doc Roadmap – Main Spec



TCG Main Spec Roadmap

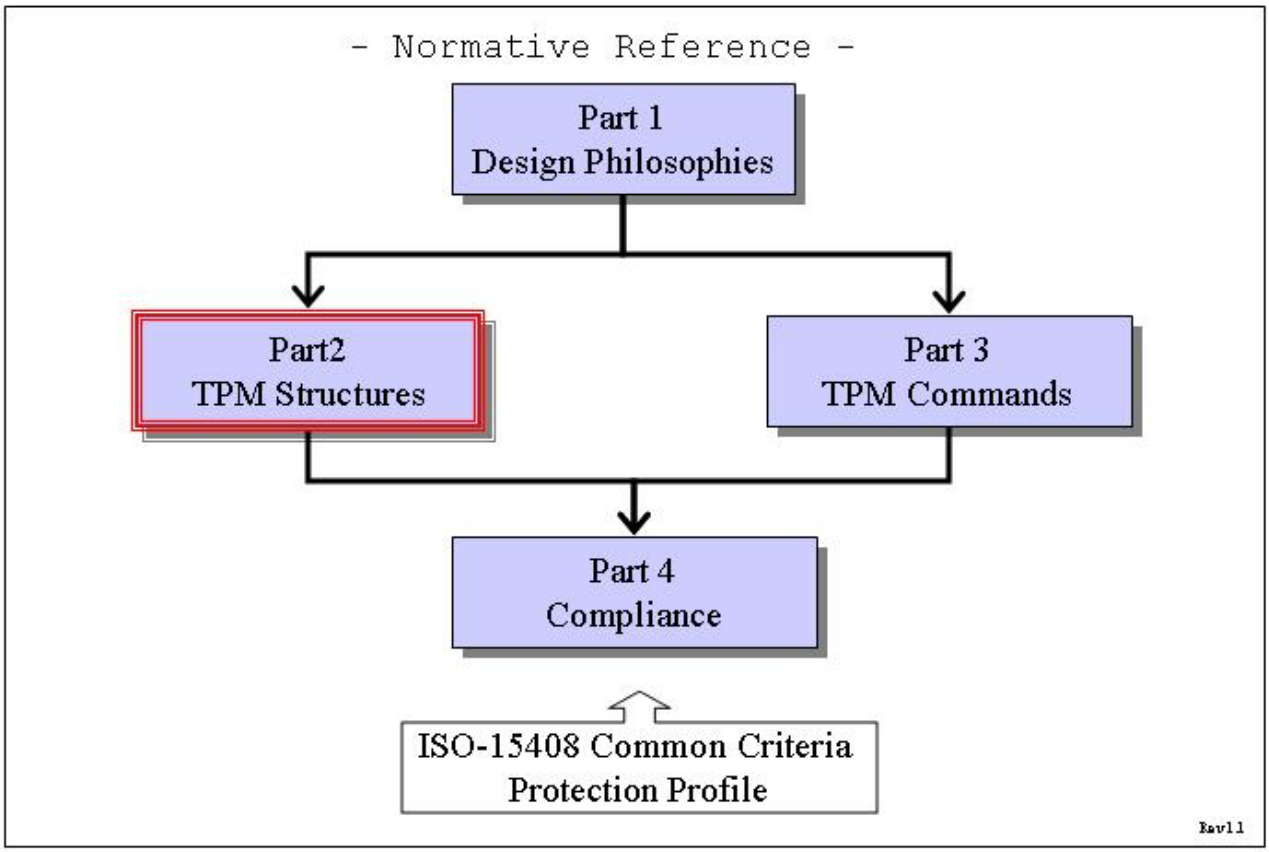


Table of Contents

End of Introduction

1. Scope and Audience

The TPCA main specification is an industry specification that enables trust in computing platforms in general. The main specification is broken into parts to make the role of each document clear. A version of the specification (like 1.2) requires all parts to be a complete specification.

This is Part 3 the structures that the TPM will use.

This document is an industry specification that enables trust in computing platforms in general.

1.1 Key words

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in the chapters 2-10 normative statements are to be interpreted as described in [RFC-2119].

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 TPM specification the user must read the specification. (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 TPM specification the user MUST read the specification. (This use of MUST indicates a keyword usage and requires an action).

2. Basic Definitions

Start of informative comment:

The following structures and formats describe the interoperable areas of the specification. 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.

2.1 Representation of Information

2.1.1 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.

2.1.2 Byte Packing

All structures MUST be packed on a byte boundary.

2.1.3 Lengths

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

2.2 Defines

Start of informative comment:

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

End of informative comment.

2.2.1 Basic data types

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

2.2.2 Boolean types

Name	Value	Description
TRUE	0x01	Assertion
FALSE	0x00	Contradiction

2.2.3 Helper redefinitions

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

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
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.
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.

Typedef	Name	Description
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 NULL, which invalidates that row. The family ID resets to NULL 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_RESTRICT_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
UINT32	TPM_GPIO_ATTRIBUTES	Attributes of the GPIO channel
UINT32	TPM_GPIO_BUS	

2.2.4 Vendor specific

Start of informative comment:

For all items that can specify an individual algorithm, protocol or item the specification 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

Name	Value
TPM_Vendor_Specific32	0x00000400
TPM_Vendor_Specific8	0x80

3. 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.

End of informative comment.

3.1 TPM_STRUCTURE_TAG

TPM_ResourceTypes

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

Name	Value	Structure
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
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_DAA_ROOTINFO	0X0030	TPM_DAA_ROOTINFO
TPM_TAG_DAA_SENSITIVE	0X0031	TPM_DAA_SENSITIVE
TPM_TAG_DAA_TPM	0X0032	TPM_DAA_TPM
TPM_TAG_GPIO_AUTHORIZE	0X0033	TPM_GPIO_AUTHORIZE
TPM_TAG_GPIO_SENSITIVE	0X0034	TPM_GPIO_SENSITIVE
TPM_TAG_GPIO_CHANNEL	0X0035	TPM_GPIO_CHANNEL

4. Types

4.1 TPM_RESOURCE_TYPE

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
	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	Handle is ignored and the resource type is pointing at the saved context blobs
	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

4.2 TPM_PAYLOAD_TYPE

Start of informative comment:

This structure specifies the type of payload in various messages.

End of informative comment.

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 asummetric key
0x07 – 0x7F		Reserved for future use by TPM
0x80 – 0xFF		Vendor specific payloads

4.3 TPM_ENTITY_TYPE

Start of informative comment:

This specifies the types of entity that are supported by the TPM.

End of informative comment.

TPM_ENTITY_TYPE Values

Value	Event Name	Key Handle	Comments
0x0001	TPM_ET_KEYHANDLE		The entity is a keyHandle
0x0002	TPM_ET_OWNER	0x40000001	The entity is the TPM Owner
0x0003	TPM_ET_DATA		The entity is some data
0x0004	TPM_ET_SRK	0x40000000	The entity is the SRK
0x0005	TPM_ET_KEY		The entity is a key
0x0006	TPM_ET_REVOKE	0x40000002	The entity is the RevokeTrust value
0x0007	TPM_ET_DEL_BLOB		The entity is a delegate blob
0x0008	TPM_ET_DEL_ROW		The entity is a delegate row
0x0009	TPM_ET_DEL_KEY		
0x000A	TPM_ET_COUNTER		The entity is a counter
0x000B	TPM_ET_NV		The entity is a NV index

4.4 Handles

Start of informative comment:

Handles provides pointers to TPM internal resources. Handles should provide the ability to locate a value without collision.

End of informative comment.

1. The TPM MAY order and set a handle to any value the TPM determines is appropriate
2. The handle value SHALL provide assurance that collisions SHOULD not occur in 2^24 handles

4.4.1 Reserved Key Handles

Start of informative comment:

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

End of informative comment.

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 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

4.5 TPM_STARTUP_TYPE

Start of informative comment:

To specify what type of startup is occurring.

End of informative comment.

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

4.6 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.

The table makeup is still an open issue.

End of informative comment.

Types of Startup

Bit position	Name	Description
31-8		No information and MUST be FALSE
7		TPM_Startup has no effect on auditDigest
6		auditDigest is set to NULL on TPM_Startup(ST_CLEAR) but not on other types of TPM_Startup
5		auditDigest is set to NULL on TPM_Startup(any)
4		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)

4.7 TPM_PROTOCOL_ID

Start of informative comment:

This value identifies the protocol in use.

End of informative comment.

Definition

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

4.8 TPM_ALGORITHM_ID

Start of informative comment:

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

End of informative comment.

TPM_ALGORITHM_ID values

Value	Name	Description
0x00000001	TPM_ALG_RSA	The RSA algorithm.
0x00000002	TPM_ALG_DES	The DES algorithm
0X00000003	TPM_ALG_3DES	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_AES128	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	TPM_ALG_AES192	AES, key size 192
0x00000009	TPM_ALG_AES256	AES, key size 256

Description

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

4.9 TPM_PHYSICAL_PRESENCE

Name	Value	Description
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

4.10 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

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 single wrapping
TPM_MS_RESTRICT_APPROVE_DOUBLE	0x0006	The key is to be migrated to an entity approved by a Migration Authority using double wrapping

4.11 TPM_EK_TYPE

Start of informative comment:

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

End of informative comment.

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

4.12 TPM_PLATFORM_SPECIFIC

Start of informative comment:

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

End of informative comment.

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	Mobil Specific version 1.2

5. Basic Structures

5.1 TPM_STRUCT_VER

Start of informative comment:

This indicates the version of the structure or TPM.

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.

End of informative comment.

IDL Definition

```
typedef struct tdTPM_STRUCT_VER {
   BYTE major;
   BYTE minor;
   BYTE revMajor;
   BYTE revMinor;
} 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
BYTE	revMinor	This MUST be 0x00

Descriptions

1. Provides the version of the structure
2. The TPM SHALL inspect all fields to determine if the TPM can properly interpret the structure.
   a. On error the TPM MUST return TPM_BAD_VERSION

5.2 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.

IDL Definition

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

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
BYTE	revMajor	This SHALL be the value of the TPM_PERSISTENT_DATA -> revMajor
BYTE	revMinor	This SHALL be the value of the TPM_PERSISTENT_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 manufactures version of the TPM
   a. Most challengers of the TPM MAY ignore the revMajor and revMinor fields

5.3 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 authorization 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 digest[digestSize];
} 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

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

5.3.1 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.

5.4 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 the specification require a nonce. This structure provides a consistent view of what a nonce is.

End of informative comment.

Definition

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

5.5 TPM_AUTHDATA

Start of informative comment:

The authorization 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 tdTPM_AUTHDATA[20];
```

Descriptions

When sending authorization 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 authorization 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

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 authorization data. The encryption mechanism depends on the context.

5.6 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.

IDL Definition

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

5.7 TPM_KEY_USAGE values

Start of informative comment:

This table defines the types of keys that are possible.

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.

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

5.7.1 Mandatory Key Usage Schemes

Start of Informative Comment:

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

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 this specification to the defined key usage values.

Name	Allowed Encryption schemes	Allowed Signature Schemes
TPM_KEY_SIGNING	TPM_ES_NONE	TPM_SS_RSASSAPKCS1v15_SHA1 TPM_SS_RSASSAPCKS1V15_DER TPM_SS_RSASSAPKCSV15_SIGNINFO
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

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.

5.8 TPM_AUTH_DATA_USAGE values

Start of informative comment:

The indication to the TPM when authorization sessions for an entity are required. The only two options at this time are always or never. Future versions may allow for more complex decisions regarding authorization checking.

End of informative comment.

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.

5.9 TPM_KEY_FLAGS

Start of informative comment:

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

End of informative comment.

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.
volatileKey	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 for commands that use the public portion of the key like TPM_GetPubKey When FALSE the TPM MUST check digestAtRelease for commands that use the public portion of the key Default value is FALSE
migrateAuthority	0x0000000C	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_MA_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

5.10 TPM_CHANGEAUTH_VALIDATE

Start of informative comment:

This structure provides an area that will stores the new authorization 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

Type	Name	Description
TPM_SECRET	newAuthSecret	This SHALL be the new authorization 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.

5.11 TPM_MIGRATIONKEYAUTH

Start of informative comment:

This structure provides the proof that the associated public key has TPM Owner authorization 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

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

5.12 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 label[4];
   TPM_ACTUAL_COUNT counter;
} 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.

5.13 TPM_SIGN_INFO Structure

Start of informative comment:

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

This is an addition in 1.2 and must be added to all commands that produce a signature. It will not be added to 1.1 commands that produce a signature.

End of informative comment.

IDL Definition

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

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

5.14 TPM_CMK_AUTH

Start of informative comment:

The signed digest of TPM_CMK_AUTH is a ticket to prove that the 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

Type	Name	Description
TPM_DIGEST	migrationAuthorityDigest	Thedigest of the public key of 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 the Migration Authority to be migrated as a child to the destinationKey.

5.15 TPM_CMK_RESTRICTDELEGATE values

Start of informative comment:

The bits of restrictMigrateDelegate are flags that determine how the TPM responds to delegated requests to manipulate a restricted-migration key.

End of informative comment.

Bit	Name	Description
31	TPM_RESTRICT_MIGRATE_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_RESTRICT_MIGRATE_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_RESTRICT_MIGRATE_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_RESTRICT_MIGRATE_LEGACY	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_LEGACY
27:0	reserved	MUST be 0

The default value of TPM_CMK_RestrictDelegate is zero (0).

6. Command 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 authorization values that follow the command parameters.

End of informative comment.

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

7. 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.

7.1 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.

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;
} TPM_PERMANENT_FLAGS;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_PERMANENT_FLAGS
BOOL	disable	The state of the disable flag. The default state is TRUE
BOOL	ownership	The ability to install an owner. The default state is TRUE.
BOOL	deactivated	The state of the inactive flag. The default state is TRUE.
BOOL	readPubek	The ability to read the PUBEK without owner authorization. The default state is TRUE.
BOOL	disableOwnerClear	Whether the owner authorized clear commands are active. The default state is FALSE.
BOOL	allowMaintenance	Whether the TPM Owner may create a maintenance archive. The default state is TRUE.
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.
BOOL	physicalPresenceHWEnable	FALSE: Disable the hardware signal indicating physical presence. (DEFAULT) TRUE: Enables the hardware signal indicating physical presence.
BOOL	physicalPresenceCMDEnable	FALSE: Disable the command indicating physical presence. (DEFAULT) TRUE: Enables the command indicating physical presence.
BOOL	CEKPUsed	TRUE: The PRIVEK and PUBEK were created using TPM_CreateEndorsementKeyPair. FALSE: The PRIVEK and PUBEK were created using a manufacturers process. NOTE: This flag has no default value as the key pair MUST be created by one or the other mechanism.

Type	Name	Description
BOOL	TPMpost	TRUE: the TPM MUST successfully complete TPM_SelfTestFull before permitting execution of any command The default state is FALSE
BOOL	TPMpostLock	FALSE: The state of TPMpost MAY be changed. (DEFAULT) TRUE: The state of TPMpost MUST NOT be changed.
BOOL	FIPS	TRUE: This TPM operates in FIPS mode FALSE: This TPM does NOT operate in FIPS mode
BOOL	operator	TRUE: The operator authorization value is valid FALSE: the operator authorization value is not set
BOOL	enableRevokeEK	TRUE: The TPM_RevokeTrust command is active FALSE: the TPM RevokeTrust command is disabled

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. TPM_Reset
    b. TPM_Init
    c. TPM_Startup
    d. TPM_SaveState
    e. TPM_SHA1Start
    f. TPM_SHA1Update
    g. TPM_SHA1Complete
    h. TPM_SHA1CompleteExtend
    i. TSC_PhysicalPresence
    j. TPM_OIAP
    k. TPM_OSAP
    l. TPM_GetCapability
    m. TPM_Extend
    n. TPM_OwnerSetDisable
    o. TPM_PhysicalEnable
    p. TPM_ContinueSelfTest
    q. TPM_SelfTestFull
    r. TPM_GetTestResult
    s. TPM_FlushSpecific
    t. TPM_PCRReset
    u. TPM_NV_Read
        i. Only values not requiring authorization

v. TPM_NV_Write

i. Only values not requiring authorization

w. All other commands SHALL return TPM_DISABLED.

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.

7.2 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 tableAdmin[TPM_MAX_FAMILY];
   BOOL bGlobalLock;
} TPM_STCLEAR_FLAGS;
```

Parameters

Type	Name	Description
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. TPM_SetTempDeactivated sets it to TRUE.
BOOL	disableForceClear	Prevents the operation of TPM_ForceClear when TRUE. The default state is FALSE. TPM_DisableForceClear sets it to TRUE.
BOOL	physicalPresence	Software indication whether an Owner is physically present. The default state is FALSE (Owner is not physically present)
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.
BOOL	tableAdmin	Value checked to determine if table admin is enabled. Reset on each TPM_Startup(ST_CLEAR)
BOOL	bGlobalLock	Set to FALSE on each TPM_Startup(ST_CLEAR). Set to TRUE when a write to NV_Index =0 is successful

Description

These values MUST reset upon execution of TPM_Startup(ST_Clear).

These values MUST NOT reset upon execution of TPM_Startup(ST_State) or TPM_Startup(ST_Deactivated)

Actions

1. If deactivated is TRUE the following commands SHALL execute with their normal protections
   a. TPM_Reset
   b. TPM_Init
   c. TPM_Startup
   d. TPM_SaveState
   e. TPM_SHA1Start
   f. TPM_SHA1Update

g. TPM_SHA1Complete
h. TPM_SHA1CompleteExtend
i. TSC_PhysicalPresence
j. TPM_OIAP
k. TPM_OSAP
l. TPM_GetCapability
m. TPM_TakeOwnership
n. TPM_OwnerSetDisable
o. TPM_PhysicalDisable
p. TPM_PhysicalEnable
q. TPM_PhysicalSetDeactivated
r. TPM_ContinueSelfTest
s. TPM_SelfTestFull
t. TPM_GetTestResult
u. TPM_FlushSpecific
v. TPM_PCRReset
w. TPM_NV_Read
    i. Only values not requiring authorization
x. TPM_NV_Write
    i. Only values not requiring authorization
y. All other commands SHALL return TPM_DEACTIVATED.

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.
5. For tableAdmin perform the following at each TPM_Startup(ST_Clear)
    a. For rowindex = 0 to NUM_FAMILY_TABLE_ENTRY
        i. Set tableAdmin[rowindex] to TPM_FAMILY_TABLE -> FamTableRow[rowindex]
    b. End for
6. Set to TRUE at TPM manufacture.

7.3 TPM_STANY_FLAGS

Start of Informative comment:

These flags reset on any TPM_Startup command.

End of informative comment.

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

Parameters

Type	Name	Description
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_MODIFIER_INDICATOR	localityModifier	This SHALL indicate for each command the presence of a locality modifier for the command. It MUST be set to NULL after the TPM executes each command.
BOOL	transportExclusive	Defaults to FALSE. TRUE when there is an exclusive transport session active. Execution of ANY command other than TPM_ExecuteTransport or TPM_ReleaseTransportSigned MUST invalidate the exclusive transport session.

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. localityModifer is set upon receipt of each command to the TPM. The localityModifier MUST be cleared when the command execution response is read

7.4 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.

IDL 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_NONCE                  tpmProof;
   TPM_NONCE                  fipsReset;
   TPM_SECRET                 ownerAuth;
   TPM_SECRET                 operatorAuth;
   TPM_SECRET                 adminAuth;
   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_TICKTYPE               tickType;
   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_RESTRICTDELEGATE   restrictDelegate;
}TPM_PERMANENT_DATA;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_PERSISTENT_DATA
BYTE	revMajor	This is the TPM major revision indicator. This SHALL be set by the TPME, only. The default value is manufacturer-specific.
BYTE	revMinor	This is the TPM minor revision indicator. This SHALL be set by the TPME, only. The default value is manufacturer-specific.
TPM_NONCE	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_SECRET	ownerAuth	This is the TPM-Owner's authorization data. The default value is manufacturer-specific.
TPM_SECRET	operatorAuth	The value that allows the execution of the SetTempDisabled command
TPM_SECRET	adminAuth	The value that allows the execution of delegation administration
TPM_TICKTYPE	tickType	This is the type of tick counter that the TPM and platform are implementing. Set once by TPM_SetTickType.
TPM_PUBKEY	manuMaintPub	This is the manufacturer's public key to use in the maintenance operations. The default value is manufacturer-specific.
TPM_KEY	endorsementKey	This is the TPM's endorsement key pair.
TPM_KEY	srk	This is the TPM's StorageRootKey.
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_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_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
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_PCR_ATTRIBUTES	pcrAttrib	The attributes for all of the PCR registers supported by the TPM.
byte	ordinalAuditStatus	Table indicating which ordinals are being audited.
TPM_DIRVALUE	authDIR	The array of TPM Owner authorized DIR. Points to the same location as the NV index value.
BYTE*	rngState	State information describing the random number generator.
TPM_FAMILY_TABLE	familyTable	The family table in use for delegations
TPM_DELEGATE_TABLE	delegateTable	The delegate table
TPM_NONCE	fipsReset	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

Type	Name	Description
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.
UINT32	TPM_LAST_FAMILYID	A value that sets the high water mark for family ID's. Set to 0 during TPM manufacturing and never reset.
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_MAXNVWRITE to any command attempting to manipulate the NV storage. Commands that manipulate the NV store are: TPM_Family_ManagetableRows TPM_Delegate_PreLoad TPM_NV_DefineSpace TPM_NV_WriteValue
TPM_CMK_RESTRICTDELEGATE	restrictDelegate	The settings that allow for the delegation and use on CMK keys

7.5 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 getcapability fields including startup effects.

End of informative comment.

IDL Definition

```
typedef struct tdTPM_STCLEAR_DATA{
   TPM_STRUCTURE_TAG    tag;
   TPM_NONCE            contextNonceKey;
   TPM_COUNT_ID         countID;
   UINT32               ownerReference;
}TPM_STCLEAR_DATA;
```

Parameters

Type	Name	Description
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 null on each TPM_Startup (ST_Clear).
TPM_COUNT_ID	countID	This is the handle for the current monotonic counter. This SHALL be set to NULL on each TPM_Startup(ST_Clear).
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.

7.6 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 getcapability fields including startup effects.

End of informative comment.

IDL Definition

```
#define TPM_MIN_SESSIONS 3

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[MAX_SESSION_LIST];
   TPM_SESSION_DATA     sessions[TPM_MIN_SESSIONS];
}TPM_STANY_DATA;
```

Parameters of STANY_Data

Type	Name	Description
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 null on each TPM_Startup (ST_Clear). The nonce MAY be set to null on TPM_Startup(any).
TPM_DIGEST	auditDigest	This is the extended value that is the audit log. This SHALL be set to NULLS at the start of each audit session.
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 the section on the tick counter for details.
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).
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). MAX_SESSION_LIST MUST be 16 or greater.

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

8. 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.

8.1 TPM_PCR_SELECTION

Start of informative comment:

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

End of informative comment.

Definition

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

When the least-significant-bit of byte [N+1] of pcrSelect is butted against the most-significant-bit of byte [N] of pcrSelect for (15>=N>=0), the contiguous bit array so formed SHALL represent PCR indices in monotonically increasing order, starting from PCR index zero represented by bit 0 of byte 0 of pcrSelect.

The state of each bit in pcrSelect indicates whether a PCR register is selected or not. When the bit is 1 then the corresponding PCR is selected, if 0 the PCR is not selected. A value of all zeros (or the selection of no PCR is valid).

pcrSelect SHALL explicitly indicate the selection or deselection of every PCR supported by the target TPM. A TPM MAY support a value of sizeOfSelect that is greater than the minimum size of pcrSelect. In v1 of the specification, this means that a TPM MUST support a sizeOfSelect greater than or equal to two.

The TPM MAY return an error if sizeOfSelect is a value other than 2

```
      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|
+--+--+--+--+--+--+--+--+
```

8.2 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

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
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

8.3 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

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.

8.4 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

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_PCR_INFO_LONG
TPM_LOCALITY_SELECTION	localityAtCreation	This SHALL be the locality modifier of the function that creates the PCR info structure
TPM_LOCALITY_SELECTION	localityAtRelease	This SHALL be the locality modifier required to reveal Sealed Data or using a key that was wrapped to PCRs
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, at the time when the sealing is performed.
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.

8.5 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.

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

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

8.6 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
```

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.

8.7 PCR Attributes

Start of informative comment:

The PCR registers will have attributes associated with the PCR register. These attributes allow for the PCR registers to be differentiated between other PCR registers.

This specification 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[15] 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[15] provide no security information. It is anticipated that PCR[15] would be set by a developer during their 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 when LOCAL_MOD is TRUE.

The pcrExtendLocal attribute modifies the PCR such that the PCR can only be Extended when LOCAL_MOD is TRUE.

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.

8.8 TPM_PCR_ATTRIBUTES

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.

End of informative comment.

IDL Definition

```
#define TPM_NUM_LOCALITY 5

typedef struct tdTPM_PCR_ATTRIBUTES{
   BOOL pcrReset;
   BOOL pcrResetLocal[TPM_NUM_LOCALITY];
   BOOL pcrExtendLocal[TPM_NUM_LOCALITY];
} 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 start9ing value MUST be 0x00..00
BOOL	pcrResetLocal	Any indicator that is set to TRUE requires that before executing the TPM_PCR_Reset command the indicated locality modifiers must be received with the command.
BOOL	pcrExtendLocal	Any indicator that is set to TRUE requires that before executing the TPM_Extend command the indicated locality modifiers must be received with the command.

8.8.1 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 V1 to TPM_VOLATILE_FLAGS

Map L1 to V1 -> localityModifier

Map P1 to TPM_PERSISTENT_DATA

If P1 -> pcrAttrib[selectedPCR].pcrExtendLocal[L1] is TRUE

return accept

else

return reject

End of informative comment.

8.9 Debug PCR register

Start of informative comment:

To accommodate debugging PCR[15] 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[15] provide no security information. It is anticipated that PCR[15] would be set by a developer during their development cycle. Developers are responsible for ensuring that a conflict between two programs does not invalidate the settings they are interested in.

End of informative comment.

The attributes for PCR[15] SHALL be the following:

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

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

PCR[15] does NOT need to be saved during TPM_SaveState

9. Storage Structures

9.1 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 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

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

9.2 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;
   UINT16 fill;
   UINT32 sealInfoSize;
   [size_is(sealInfoSize)] BYTE* sealInfo;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE* encData;
} TPM_STORED_DATA12;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_STORED_DATA12
UINT16	fill	MUST be 0x0000
UINT32	sealInfoSize	Size of the sealInfo parameter
BYTE*	sealInfo	This SHALL be a structure of type TPM_PCR_INFO_LONG 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. If sealInfoSize is not 0 then sealInfo MUST be TPM_PCR_INFO_LONG

9.3 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_NONCE tpmProof;
   TPM_DIGEST storedDigest;
   UINT32 dataSize;
   [size_is(dataSize)] BYTE* data;
} 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 authorization data for this value
TPM_NONCE	tpmProof	This SHALL be a copy of TPM_PERMANENT_FLAGS -> 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.

9.4 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

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

9.5 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 TCG 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 maximise 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

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_M). If it is not, the TPM_UnBind command will fail.

10. 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.:

10.1 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

Type	Name	Description
TPM_ALGORITHM_ID	algorithmID	This SHALL be the key algorithm in use
UINT32	parmSize	This SHALL be the size of the parms field in bytes
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
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:

Algorithm Id	PARMS Contents
TPM_ALG_RSA	A structure of type TPM_RSA_KEY_PARMS
TPM_ALG_DES	A structure of type TPM_SYMMETRIC_KEY_PARMS
TPM_ALG_3DES	A structure of type TPM_SYMMETRIC_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

10.1.1 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

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

10.1.2 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

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

10.2 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;
   BYTE* PCRInfo;
   TPM_STORE_PUBKEY     pubKey;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE* encData;
} 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 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

10.3 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;
   BYTE* PCRInfo;
   TPM_STORE_PUBKEY     pubKey;
   UINT32 encDataSize;
   [size_is(encDataSize)] BYTE* encData;
} 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, 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 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

10.4 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

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:

Algorithm Id	'Key' Contents
TPM_ALG_RSA	The RSA public modulus

10.5 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 it's 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

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.

10.6 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 10.8).

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

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL set to TPM_PT_ASYM to indicate an asymmetric key.
TPM_SECRET	usageAuth	This SHALL be the authorization data necessary to authorize the use of this value
TPM_SECRET	migrationAuth	This SHALL be the migration authorization 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 authorization 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 calcuation.
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.

10.7 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

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.

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.

10.8 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 TPM's.

This structure provides the basis of defining the protection of the private key. For the complete description of the entire encryption process, see **Error! Reference source not found.**.

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
   TPM_STORE_PRIVKEY partPrivKey;          //  45   112-127  157-172
} TPM_MIGRATE_ASYMKEY;
```

Parameters

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL set to TPM_PT_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
TPM_STORE_PRIVKEY	partPrivKey	This SHALL be the k2 area as

10.9 TPM_KEY_CONTROL

Start of informative comment:

Attributes that can control various aspects of key usage and manipulation

End of informative comment.

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.

11. Signed Structures

11.1 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 null to avoid collisions with the other signature headers.

End of informative comment.

IDL 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

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 NULL
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 NULL.
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.

11.2 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 null to avoid collisions with the other signature headers.

End of informative comment.

IDL Definition

```
typedef struct tdTPM_CERTIFY_INFO2{
   TPM_STRUCTURE_TAG tag;
   UINT16 fill;
   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;
   UNIT32 migrationAuthoritySize ;
   [size_is(migrationAuthoritySize)] BYTE migrationAuthority;
   UINT32 PCRInfoSize;
   [size_is(pcrInfoSize)] BYTE* PCRInfo;
} TPM_CERTIFY_INFO2
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CERTIFY_INFO2
UINT16	fill	MUST be 0x0000
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 NULL
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 NULL.
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	migrationAuthoritySize	This SHALL be the size of migrationAuthority
BYTE[]	migrationAuthority	If the key to be certified has [payload == TPM_PT_MIGRATE_RESTRICTED], migrationAuthority is the digest of the TPM_PUBKey of the key's migrationAuth and has TYPE == TPM_DIGEST. Otherwise it is NULL
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_SHORT structure.

11.3 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.

IDL Definition

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

Parameters

Type	Name	Description
TPM_STRUCT_VER	version	This MUST be 1.1.0.0
BYTE	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

12. Identity Structures

12.1 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

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

12.2 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

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 identity public key that is being certified by the CA
TPM_PCR_INFO_SHORT	pcrInfo	This SHALL indicate the PCR's and localities
TPM_COMPOSITE_HASH	compositeHash	This SHALL be the value of the indicated PCR registers

12.3 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

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

12.4 TPM_CHOSENID_HASH

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

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

1. TPM_CHOSENID_HASH = SHA(identityLabel || privacyCA)

12.5 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.

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

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
UINT32	rdinal	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

12.6 TPM_IDENTITY_REQ

Start of informative comment:

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

End of informative comment.

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

12.7 TPM_IDENTITY_PROOF

Start of informative comment:

Structure in use during the AIK credential process.

End of informative comment.

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

12.8 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

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 identity public key that is being certified by the CA

12.9 TPM_SYM_CA_ATTESTATION

Start of informative comment:

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

End of informative comment.

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"

13. Transport structures

13.1 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

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.

13.1.1 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

13.2 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 tranPublic;
   TPM_TRANSHANDLE transHandle;
   TPM_NONCE transEven;
   TPM_DIGEST transDigest;
} 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	tranPublic	The public information of this session
TPM_TRANSHANDLE	transHandle	The handle for this session
TPM_NONCE	transEven	The even nonce for the rolling protocol
TPM_DIGEST	transDigest	The log of transport events

13.3 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.

IDL Definition

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

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_LOG_IN
TPM_COMMAND_CODE	Ordinal	Ordinal of the command
TPM_DIGEST	parameters	When computed as part of the TPM_EstablishTransport command and as part of the TPM_ReleaseTransportSigned command, the value of this parameter is identical to the SHA-1 hash value used as part of the normal authorization computation. When computed as part of TPM_ExecuteTransport, then following rules are used: SHA1(ORDw, DATAINw) where DATAINw is the concatenation of ingoing wrapped command parameters excluding handles.
TPM_DIGEST	pubKeyHash	The hash of any keys in the transport command

13.4 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.

IDL Definition

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

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. The value is set to 0 on input to ExecuteTransport to avoid timing attacks.
TPM_DIGEST	parameters	When computed as part of the TPM_EstablishTransport command and as part of the TPM_ReleaseTransportSigned command, the value of this parameter is identical to the SHA-1 hash value used as part of the normal authorization computation. When computed as part of TPM_ExecuteTransport, then following rules are used: SHA1(RCw,ORDw,DATAOUTw) where DATAOUTw is the concatenation of outgoing wrapped command parameters excluding handles.

13.5 TPM_TRANSPORT_AUTH structure

Start of informative comment:

This structure provides the validation for the encrypted authorization value.

End of informative comment.

IDL Definition

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

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_AUTH
TPM_AUTHDATA	authData	The authorization value

14. Audit Structures

14.1 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.

IDL Definition

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

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_AUDIT_EVENT_IN
TPM_COMMAND_CODE	ordinal	Ordinal of the command
TPM_DIGEST	inputParms	Digest value according to the HMAC digest rules.
TPM_COUNTER_VALUE	auditCount	The current value of the audit monotonic counter

14.2 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.

IDL Definition

```
typedef struct tdTPM_AUDIT_EVENT_OUT {
   TPM_STRUCTURE_TAG    tag;
   TPM_COMMAND_CODE ordinal;
   TPM_DIGEST outputParms;
   TPM_COUNTER_VALUE auditCount;
   TPM_RESULT returncode;
} TPM_AUDIT_EVENT_OUT;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_AUDIT_EVENT_OUT
TPM_COMMAND_CODE	ordinal	Ordinal of the command
TPM_DIGEST	outputParms	Digest value according to the HMAC digest rules.
TPM_COUNTER_VALUE	auditCount	The current value of the audit monotonic counter
TPM_RESULT	returncode	Return code for the command

15. Tick Structures

15.1 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 relationship between ticks and seconds. The <tickType> element of TPM_CURRENT_TICKS structure provides the definitions for the events which the start of a clock session

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.

IDL Definition

```
typedef struct tdTPM_CURRENT_TICKS {
   TPM_STRUCTURE_TAG    tag;
   UINT64 currentTicks;
   UINT16   tickType;
   UINT16   tickRate;
   UINT16   tickSecurity;
   TPM_NONCE      tickNonce;
}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	tickType	The implementation model of the clock
UINT16	tickRate	One tick represents x microseconds. The maximum resolution of the TPM tick counter would then be 1 microsecond. The minimum resolution SHOULD be 1 millisecond.
UINT16	tickSecurity	The security model for the clock
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.

15.1.1 TPM_TICKTYPE values

Start of informative comment:

These values indicate to challengers of the TPM the mechanism that the TPM and the platform use to maintain the tick values inside of the TPM.

End of informative comment.

Name	Value	Description

TICK_INC	0x00	The clock is implemented as a counter that increments at a maximum rate of <tickRate> when the system is running. Depending on the system implementation, the clock may increment at an arbitrarily slower rate. The TPM guarantees that the clock value will increment at least once prior to the execution of any command. Execution of TPM_Startup (ST_CLEAR or ST_STATE) reflects a loss of the counter value.
TICK_INC_SAVE	0x01	As TICK_INC except that the current time is saved (but not incremented) across intervals when the ability to increment is lost.
TICK_POWER	0x02	The clock is implemented as a counter that increments at the rate of <tickRate> when the system is running. Power may be available between execution of TPM_SaveState and TPM_Startup(ST_STATE). When power is lost execution of TPM_Startup will reflect the loss of counter value.
TICK_POWER_SAVE	0x03	As TICK_POWER except that the current time is saved (but not incremented) across intervals when the ability to increment is lost.
TICK_STSTATE	0x04	The clock is implemented as a counter that increments at the rate of <tickRate> when the system is running. The clock also increments from the time of TPM_SaveState to the execution of TPM_Startup(ST_STATE). Execution of TPM_Startup (ST_CLEAR) reflects a loss of counter value.
TICK_STSTATE_SAVE	0x05	As TICK_STSTATE except that the current time is saved (but not incremented) across intervals when the ability to increment is lost.
TICK_STCLEAR	0x06	The clock is implemented as a counter that increments at the rate of <tickRate> when the system is running. The clock also continues to increment between TPM_Startup(ST_CLEAR or ST_STATE). The TPM MAY lose the ability to increment between these events and if so then the time value is lost.
TICK_STCLEAR_SAVE	0x07	As TICK_STCLEAR except that the current time is saved (but not incremented) across intervals when the ability to increment is lost.
TICK_ALWAYS	0x08	The clock is implemented as a counter that increments at the rate of <tickRate> regardless of the state of the system.

Start of informative comment:

The following are some PC implementations that give the flavor of what is possible regarding the clock on a specific platform.

TICK_INC No TPM power battery. Clock comes from PCI clock, may stop from time to time due to clock stopping protocols such as CLKRUN.

TICK_POWER No TPM power battery. Clock source comes from PCI clock, always runs except in S3+.

TICK_STSTATE External power (might be battery) consumed by TPM during S3 only. Clock source comes either from a system clock that runs during S3 or from crystal/internal TPM source.

TICK_STCLEAR Standby power used to drive counter. In desktop, may be related to when system is plugged into wall. Clock source comes either from a system clock that runs when standby power is available or from crystal/internal TPM source.

TICK_ALWAYS TPM power battery. Clock source comes either from a battery powered system clock that crystal/internal TPM source.

TICK_XXX_SAVE External power (might be battery) maintains state when the TPM is not counting.

End of informative comment.

15.1.2 TickSecurity Values

Name	Value	Description
NO_CHECK	0x00	The TPM does not implement any internal hardware security checks to verify that the tick value is valid. Assurance MUST come from the system design and the system evaluation.
RATE_CHECK	0x01	The TPM implements an internal hardware security check to ensure that the tick counter accurately reflects the number of ticks since the start of the tick session.

16. 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), 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 a non-authenticated fixed set of information, only. This is due to the fact that 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 and continue the nonce's 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. TPM_TAG_RQU_COMMAND (2 bytes)
   b. ParamLength(4 bytes, fixed at 10)
   c. Return Code (4 bytes, never TPM_SUCCESS)
2. When a capability has failed to complete successfully, 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 capability, it SHOULD be the error code specified by the capability 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 TCG defined or vendor defined.

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

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 presented was invalid
TPM_KEYNOTFOUND	TPM_BASE + 13	The target key was not found
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_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
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, phsicalPresence 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 writtable
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
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 authorization 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_EXCLUSIVE	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 availble to execute the command

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

17. Ordinals

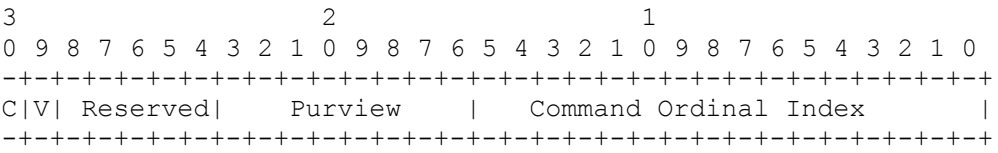
Start of informative comment:

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.



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

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 main specification. Unprotected connection commands are specified in the TSS
0x20000000	TPM_VENDOR_COMMAND	Command that is vendor specific for a given TPM or TSS.

The following Purviews have been defined:

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

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.

Blank is not supported

X means column is active

D deleted

N never

	TPM_PROTECTED_ORDINAL	Complete ordinal	AUTH2	AUTH1	RQU	Optional	Deprecated	No Owner	PCR Use	Audit
TPM_ORD_ActivateIdentity	122	0x0000007A	x	x					x	x
TPM_ORD_AuthorizeMigrationKey	43	0x0000002B		x						x
TPM_ORD_CertifyKey	50	0x00000032	x	x	x				x	
TPM_ORD_CertifyKey2	51	0x00000033	x	x	x				x	
TPM_ORD_CertifySelfTest	82	0x00000052		x	x		x			
TPM_ORD_ChangeAuth	12	0x0000000C	x							
TPM_ORD_ChangeAuthAsymFinish	15	0x0000000F		x	x		x			
TPM_ORD_ChangeAuthAsymStart	14	0x0000000E		x	x		x			
TPM_ORD_ChangeAuthOwner	16	0x00000010		x						x
TPM_ORD_CMK_CreateBlob	27	0x0000001B		x						
TPM_ORD_CMK_CreateKey	19	0x00000013		x						
TPM_ORD_CMK_CreateTicket	18	0x00000012		x						
TPM_ORD_CMK_SetRestrictions	28	0x0000001C		x						
TPM_ORD_ContinueSelfTest	83	0x00000053			x			x		
TPM_ORD_ConvertMigrationBlob	42	0x0000002A		x	x				x	x

TPM_ORD_CreateCounter	220	0x000000DC		x						
TPM_ORD_CreateEndorsementKeyPair	120	0x00000078			x			x		
TPM_ORD_CreateMaintenanceArchive	44	0x0000002C		x		x				x
TPM_ORD_CreateMigrationBlob	40	0x00000028	x	x					x	x
TPM_ORD_CreateRevocableEK	127	0x0000007F			x	x		x		
TPM_ORD_CreateWrapKey	31	0x0000001F		x					x	x
TPM_ORD_Delegate_CreateKeyDelegation	212	0x000000D4			x			x		
TPM_ORD_Delegate_CreateOwnerDelegation	213	0x000000D5		x	x					
TPM_ORD_Delegate_LoadOwnerDelegation	216	0x000000D8		x	x					
TPM_ORD_Delegate_Manage	210	0x000000D2		x	x			x		
TPM_ORD_Delegate_ReadAuth	217	0x000000D9			x					
TPM_ORD_Delegate_ReadTable	219	0x000000DB			x			x		
TPM_ORD_Delegate_UpdateVerification	209	0x000000D1		x						
TPM_ORD_Delegate_VerifyDelegation	214	0x000000D6			x					
TPM_ORD_DirRead	26	0x0000001A			x		x			
TPM_ORD_DirWriteAuth	25	0x00000019		x			x			
TPM_ORD_DisableForceClear	94	0x0000005E		x				x		x
TPM_ORD_DisableOwnerClear	92	0x0000005C		x						x
TPM_ORD_DisablePubekRead	126	0x0000007E		x						x
TPM_ORD_DSAP	17	0x00000011			x				x	
TPM_ORD_EstablishTransport	230	0x000000E6		x						
TPM_ORD_EvictKey	34	0x00000022					x			
TPM_ORD_ExecuteTransport	231	0x000000E7		x						
TPM_ORD_Extend	20	0x00000014			x			x		
TPM_ORD_FieldUpgrade	170	0x000000AA	x	x	x	x		x		
TPM_ORD_FlushSpecific	186	0x000000BA			x			x		
TPM_ORD_ForceClear	93	0x0000005D			x			x		x
TPM_ORD_GetAuditDigest	133	0x00000085			x			x		N
TPM_ORD_GetAuditDigestSigned	134	0x00000086		x	x					N
TPM_ORD_GetAuditEvent	130	0x00000082			x	x	D			N
TPM_ORD_GetAuditEventSigned	131	0x00000083		x	x	x	D			N
TPM_ORD_GetCapability	101	0x00000065			x			x		
TPM_ORD_GetCapabilityOwner	102	0x00000066		x			D			
TPM_ORD_GetCapabilitySigned	100	0x00000064		x	x		D			
TPM_ORD_GetOrdinalAuditStatus	140	0x0000008C			x		D			N
TPM_ORD_GetPubKey	33	0x00000021		x	x				x	
TPM_ORD_GetRandom	70	0x00000046			x			x		
TPM_ORD_GetTestResult	84	0x00000054			x			x		
TPM_ORD_GetTick	241	0x000000F1			x			x		
TPM_ORD_IncrementCounter	221	0x000000DD		x						
TPM_ORD_Init	151	0x00000097			x					

TPM_ORD_KeyControlOwner	35	0x00000023		x						
TPM_ORD_KillMaintenanceFeature	46	0x0000002E		x		x				x
TPM_ORD_LoadAuthContext	183	0x000000B7			x	x	x	x		
TPM_ORD_LoadContext	185	0x000000B9			x			x		
TPM_ORD_LoadKey	32	0x00000020		x	x					
TPM_ORD_LoadKeyContext	181	0x000000B5			x	x	x	x		
TPM_ORD_LoadMaintenanceArchive	45	0x0000002D		x		x				x
TPM_ORD_LoadManuMaintPub	47	0x0000002F				x				x
TPM_ORD_MakeIdentity	121	0x00000079	x	x						x
TPM_ORD_NV_DefineSpace	204	0x000000CC		x	x			x		
TPM_ORD_NV_ReadValue	207	0x000000CF		x	x			x	x	
TPM_ORD_NV_ReadValueAuth	208	0x000000D0		x					x	
TPM_ORD_NV_WriteValue	205	0x000000CD		x	x			x	x	
TPM_ORD_NV_WriteValueAuth	206	0x000000CE		x					x	
TPM_ORD_OIAP	10	0x0000000A			x			x		
TPM_ORD_OSAP	11	0x0000000B			x				x	
TPM_ORD_OwnerClear	91	0x0000005B		x						x
TPM_ORD_OwnerReadInternalPub	129	0x00000081		X						
TPM_ORD_OwnerReadPubek	125	0x0000007D		x			x			x
TPM_ORD_OwnerSetDisable	110	0x0000006E		x						x
TPM_ORD_PCR_Reset	200	0x000000C8			x			x		
TPM_ORD_PcrRead	21	0x00000015			x			x		
TPM_ORD_PhysicalDisable	112	0x00000070			x			x		x
TPM_ORD_PhysicalEnable	111	0x0000006F			x			x		x
TPM_ORD_PhysicalSetDeactivated	114	0x00000072			x			x		x
TPM_ORD_Quote	22	0x00000016		x	x				x	
TPM_ORD_ReadCounter	222	0x000000DE		x	x			x		
TPM_ORD_ReadManuMaintPub	48	0x00000030				x				x
TPM_ORD_ReadPubek	124	0x0000007C			x			x		x
TPM_ORD_ReleaseCounter	223	0x000000DF		x				x		
TPM_ORD_ReleaseCounterOwner	224	0x000000E0		x						
TPM_ORD_ReleaseTransportSigned	232	0x000000E8		x					x	
TPM_ORD_Reset	90	0x0000005A			x		x	x		
TPM_ORD_RevokeTrust	128	0x00000080		x		x		x		
TPM_ORD_SaveAuthContext	182	0x000000B6			x	x	x	x		
TPM_ORD_SaveContext	184	0x000000B8			x			x		
TPM_ORD_SaveKeyContext	180	0x000000B4			x	x	x	x		
TPM_ORD_SaveState	152	0x00000098			x			x		
TPM_ORD_Seal	23	0x00000017		x						
TPM_ORD_SelfTestFull	80	0x00000050			x			x		
TPM_ORD_SetOperatorAuth	116	0x00000074		x				x		

TPM_ORD_SetOrdinalAuditStatus	141	0x0000008D		x			D			x
TPM_ORD_SetOwnerInstall	113	0x00000071		x				x		x
TPM_ORD_SetOwnerPointer	117	0x00000075			x					
TPM_ORD_SetRedirection	154	0x0000009A			x	x				x
TPM_ORD_SetTempDeactivated	115	0x00000073		x				x		x
TPM_ORD_SetTickType	240	0x000000F0			x			x		
TPM_ORD_SHA1Complete	162	0x000000A2			x			x		
TPM_ORD_SHA1CompleteExtend	163	0x000000A3			x			x		
TPM_ORD_SHA1Start	160	0x000000A0			x			x		
TPM_ORD_SHA1Update	161	0x000000A1			x			x		
TPM_ORD_Sign	60	0x0000003C		x	x				x	
TPM_ORD_Startup	153	0x00000099			x			x		
TPM_ORD_StirRandom	71	0x00000047			x			x		
TPM_ORD_TakeOwnership	13	0x0000000D		x				x		x
TPM_ORD_Terminate_Handle	150	0x00000096			x		x	x		
TPM_ORD_TickStampBlob	242	0x000000F2		x	x					
TPM_ORD_UnBind	30	0x0000001E		x	x					
TPM_ORD_Unseal	24	0x00000018	x	x					x	
UNUSED	29	0x0000001D								
UNUSED	36	0x00000024								
UNUSED	37	0x00000025								
UNUSED	38	0x00000026								
UNUSED	39	0x00000027								
UNUSED	41	0x00000029								
UNUSED	49	0x00000031								
UNUSED	61	0x0000003D								
UNUSED	62	0x0000003E								
UNUSED	63	0x0000003F								
UNUSED	64	0x00000040								
UNUSED	65	0x00000041								
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								
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								
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	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	201	0x00000C9								

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

	TPI	Comp I	A1	A1	RC	O	D	N	PC	A
TSC_OR_D_RESET TSC_OR_D_PhysicalPresence Establishment Bit	10	0x40000000 0x40000000 0A			x			x		
	11	0x40000000 0B			x					

18. Context structures

18.1 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 in use 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.

End of informative comment.

IDL Definition

```
typedef struct tdTPM_CONTEXT_BLOB {
   TPM_STRUCTURE_TAG tag;
   TPM_RESOURCE_TYPE resourceType;
   TPM_HANDLE handle;
   BYTE[16] label;
   UINT32 contextCount;
   TPM_DIGEST blobIntegrity;
   UINT32 additionalSize;
   [size_is(additionalSize)] BYTE* additionalData;
   UINT32 sensitiveSize;
   [size_is(sensitiveSize)] BYTE* sensitiveData;
}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_VOLATILE_DATA -> contextCount when creating the structure. This value is ignored for context blobs that reference a key.
TPM_DIGEST	blobIntegrity	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 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

18.2 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.

IDL Definition

```
typedef struct tdTPM_CONTEXT_SENSITIVE {
   TPM_STRUCTURE_TAG tag;
   TPM_NONCE contextNonce;
   UINT32 internalSize;
   [size_is(internalSize)] BYTE* internalData;
}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_VOLATILE_DATA -> contextNonceSession For keys the value is TPM_VOLATILE_DATA -> contextNonceKey
UINT32	internalSize	The size of the internalData area
BYTE	internalData	The internal data area

19. NV storage structures

19.1 TPM_NV_INDEX

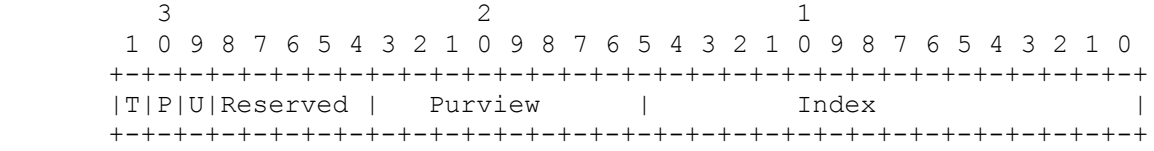
Start of informative comment:

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 TCG 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.



Where:

1. All reserved area bits are set to 0
   a. T is the TPM manufacturer reserved bit. 0 indicates TCG 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. TCG reserved areas have T/P/U set to 0.
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.

19.1.1 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_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

Value	Index Name	Default Size	Attributes
0xFFFFFFFFFFFFFFFF	TPM_NV_INDEX_LOCK	Size for this MUST be 0. This value turns on the NV authorization protections. Once executed all NV areas us the protections as defined. This value never resets	None
0x0000000000	TPM_NV_INDEX0	Size for this MUST be 0. This value allows for the setting of the persistent lock bit which is only reset on TPM_Startup(ST_Clear)	None
0x0000000001	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	TPM_NV_PER_OWNERWRITE TPM_NV_PER_WRITEALL

		on the the 1.1 DIR commands.	

19.1.2 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_DefineSpace to have the area for the index allocated
3. A platform specific specification MAY indicate that reserved values are required.

Value	Event Name	Default Size
0x000000F000	TPM_NV_INDEX_EKCert	The Endorsement credential
0x000000F001	TPM_NV_INDEX_TPM_CC	The TPM Conformance credential
0x000000F002	TPM_NV_INDEX_PlatformCert	The platform credential
0x000000F003	TPM_NV_INDEX_Platform_CC	The Platform conformance credential

19.2 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.

End of informative comment.

Definition

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

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_NV_ATTRIBUTES
UINT32	attributes	The attribute area

Attributes values

Bit	Name	Description
31	TPM_NV_PER_READ_STCLEAR	The value can only be read once per TPM_Startup(ST_Clear) cycle. Lock set by a read with a datasize of 0
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 writeable until a write to index 0 is successful. The lock of this attribute is reset by TPM_Startup(ST_CLEAR). Lock held by SV -> bGlobalLock
14	TPM_NV_PER_WRITE_STCLEAR	The value is writeable 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	The value can only be written once after performing the TPM_NV_DefineSpace command. Lock held for each area as bWriteDefine. Lock set by writing to the index with a datasize of 0
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

19.3 TPM_NV_DATA_PUBLIC

Start of informative comment:

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

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

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL 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_ATTRIBUTESS	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 WriteValue with a datasize of 0
UINT32	DataSize	The size of the data area in bytes

19.4 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_AUTH authValue;
   [size_is(dataSize)] BYTE* data;
} TPM_NV_DATA_SENSITIVE;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_NV_DATA_SENSITIVE
TPM_NV_DATA_PUBLIC	pubInfo	The public information regarding this area
TPM_AUTH	authValue	The authorization 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.

20. Delegate Structures

20.1 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 TPM's 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 or a full three key triple DES.

20.2 Delegate Definitions

Informative comment:

The delegations are in a 64-bit field. Each bit describes a capability that the TPM Owner 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 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

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

20.2.1 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

Bit Number	Ordinal	Bit Name
31:27	Reserved	Reserved MUST be 0
26	TPM_ORD_CMD_CreateTicket	TPM_DELEGATE_CMD_CreateTicket
25	TPM_ORD_CMK_CreateKey	TPM_DELEGATE_CMK_CreateKey
24	TPM_ORD_Delegate_LoadOwnerDelegation	TPM_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_Delegate_CreateKeyDelegation	TPM_DELEGATE_CreateKeyDelegation
17	TPM_ORD_LoadBlobOwner	TPM_Delegate_LoadOwnerDelegation
16	TPM_ORD_OwnerClear	TPM_DELEGATE_OwnerClear
15	TPM_ORD_DisableOwnerClear	TPM_DELEGATE_DisableOwnerClear
14	TPM_ORD_DisableForceClear	TPM_DELEGATE_DisableForceClear
13	TPM_ORD_OwnerSetDisable	TPM_DELEGATE_OwnerSetDisable
12	TPM_ORD_SetOwnerInstall	TPM_DELEGATE_SetOwnerInstall
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_ 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

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

20.2.2 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.

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 owners 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

20.3 Key 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

Bit Number	Ordinal	Bit Name
31:11	Reserved	Reserved MUST be 0
10	TPM_ORD_CMK_CreateBlob	TPM_DELEGATE_CMK_CreateBlob
9	TPM_ORD_CreateMigrationBlob	TPM_DELEGATE_CreateMigrationBlob
8	TPM_ORD_ConvertMigrationBlob	TPM_DELEGATE_ConvertMigrationBlob
7	TPM_ORD_CreateBlob	TPM_Delegate_CreateKeyDelegation
6		
5	TPM_ORD_GetPubKey	TPM_DELEGATE_GetPubKey
4	TPM_ORD_Unbind	TPM_DELEGATE_Unbind
3	TPM_ORD_Quote	TPM_DELEGATE_Quote
2	TPM_ORD_Unseal	TPM_DELEGATE_Unseal
1	TPM_ORD_Seal	TPM_DELEGATE_Seal
0	TPM_ORD_LoadKey	TPM_DELEGATE_LoadKey

Per2 bits

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

20.3.1 Key commands not delegated

Start of informative comment:

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

End of informative comment.

Command	Rationale
None	

20.4 TPM_FAMILY_FLAGS

Start of informative comment:

These flags indicate the operational state of the delegation and family table. These flags are additions to TPM_PERMANENT_FLAGS and are not standalone values.

End of informative comment.

TPM_FAMILY_FLAGS bit settings

Bit Number	Bit Name	Comments
31:2	Reserved MUST be 0	
1	DELEGATE_ADMIN_LOCK	TRUE: Some TPM_Delegate_XXX commands are locked and return TPM_DELEGATE_LOCK FALSE: TPM_Delegate_XXX commands are available Default is FALSE
0	TPM_FAMFLAG_ENABLE	When TRUE the table is enabled. The default vaue is FALSE.

20.5 TPM_FAMILY_LABEL

Start of informative comment:

Used in the family table to hold a one-byte numeric value (sequence number) that software can map to a string of bytes that can be displayed or used by applications.

This is not sensitive data.

End of informative comment.

```
typedef struct tdTPM_FAMILY_LABEL{
   BYTE label;
} TPM_FAMILY_LABEL;
```

Parameters

Type	Name	Description
BYTE	label	A sequence number that software can map to a string of bytes that can be displayed or used by the applications. This MUST not contain sensitive information.

20.6 TPM_FAMILY_TABLE_ENTRY

Start of informative comment:

The family table entry is an individual row in the family table. There are no sensitive values in a family table entry.

Each family table entry contains values to facilitate table management: the familyID sequence number value that associates a family table row with one or more delegate table rows, a verification sequence number value that identifies when rows in the delegate table were last verified, and a 32-bit numeric family label value that software can map to an ASCII text description of the entity using the family table entry

End of informative comment.

```
typedef struct tdTPM_FAMILY_TABLE_ENTRY{
   TPM_STRUCTURE_TAG tag;
   TPM_FAMILY_LABEL familyLabel;
   TPM_FAMILY_ID familyID;
   TPM_FAMILY_VERIFICATION verificationCount;
   TPM_FAMILY_FLAGS flags;
} TPM_FAMILY_TABLE_ENTRY;
```

Description

The default value of all fields in a family row at TPM manufacture SHALL be null.

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_FAMILY_TABLE_ENTRY
TPM_DELEGATE_LABEL	familyLabel	The ASCII text description of the entity using this family row. This is not a sensitive value.
TPM_FAMILY_ID	familyID	The family ID in use to tie values together. This is not a sensitive value.
TPM_FAMILY_VERIFICATION	verificationCount	The value inserted into delegation rows to indicate that they are the current generation of rows. Used to identify when a row in the delegate table was last verified. This is not a sensitive value.
TPM_FAMILY_FLAGS	flags	See section on TPM_FAMILY_FLAGS.

20.7 TPM_FAMILY_TABLE

Start of informative comment:

The family table is stored in a TPM shielded location. There are no confidential values in the family table. The family table contains a minimum of 8 rows.

End of informative comment.

```
#define TPM_NUM_FAMILY_TABLE_ENTRY_MIN 8

typedef struct tdTPM_FAMILY_TABLE{
   TPM_FAMILY_TABLE_ENTRY FamTableRow[TPM_NUM_FAMILY_TABLE_ENTRY_MIN];
} TPM_FAMILY_TABLE;
```

Parameters

Type	Name	Description
TPM_FAMILY_TABLE_ENTRY	FamTableRow	The array of family table entries

20.8 TPM_DELEGATE_LABEL

Start of informative comment:

Used in both the delegate table and the family table to hold a string of bytes that can be displayed or used by applications. This is not sensitive data.

End of informative comment.

```
typedef struct tdTPM_DELEGATE_LABEL{
   BYTE label;
} TPM_DELEGATE_LABEL;
```

Parameters

Type	Name	Description
BYTE	label	A byte that can be displayed or used by the applications. This MUST not contain sensitive information.

20.9 TPM_DELEGATE_PUBLIC

Start of informative comment:

The information of a delegate row that is public and does not have any sensitive information.

PCR_INFO_SHORT is appropriate here as the command to create this is done using owner authorization, hence the owner authorized the command and the delegation. There is no need to validate what configuration was controlling the platform during the blob creation.

End of informative comment.

```
typedef struct tdTPM_DELEGATE_PUBLIC{
   TPM_STRUCTURE_TAG tag;
   TPM_DELEGATE_LABEL rowLabel;
   TPM_PCR_INFO_SHORT pcrInfo;
   TPM_DELEGATIONS permissions;
   TPM_FAMILY_ID familyID;
   TPM_FAMILY_VERIFICATION verificationCount
} TPM_DELEGATE_PUBLIC;
```

Description

The default value of all fields of a delegate row at TPM manufacture SHALL be null. The table MUST NOT contain any sensitive information.

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_DELEGATE_PUBLIC
TPM_DELEGATE_LABEL	rowlabel	This SHALL be the label for the row. It MUST not contain any sensitive information.
TPM_PCR_INFO_SHORT	pcrInfo	This SHALL be the designation of the process that can use the permission. This is a not sensitive value. PCR_SELECTION may be NULL. If selected the pcrInfo MUST be checked on each use of the delegation. Use of the delegation is where the delegation is passed as an authorization handle.
TPM_DELEGATIONS	permissions	This SHALL be the permissions that are allowed to the indicated process. This is not a sensitive value.
TPM_FAMILY_ID	familyID	This SHALL be the family ID that identifies which family the row belongs to. This is not a sensitive value.
TPM_FAMILY_VERIFICATION	verificationCount	A copy of verificationCount from the associated family table. This is not a sensitive value.

20.10 TPM_DELEGATE_TABLE_ROW

Start of informative comment:

A row of the delegate table.

End of informative comment.

```
typedef struct tdTPM_DELEGATE_TABLE_ROW{
   TPM_STRUCTURE_TAG tag;
   TPM_DELEGATE_PUBLIC pub;
   TPM_SECRET authValue;
} TPM_DELEGATE_TABLE_ROW;
```

Description

The default value of all fields of a delegate row at TPM manufacture SHALL be empty

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_DELEGATE_TABLE_ROW
TPM_DELEGATE_PUBLIC	pub	This SHALL be the public information for a table row.
TPM_SECRET	authValue	This SHALL be the authorization value that can use the permissions. This is a sensitive value.

20.11 TPM_DELEGATE_TABLE

Start of informative comment:

This is the delegate table. The table contains a minimum of 2 rows.

This will be an entry in the TPM_PERSISTENT_DATA structure.

End of informative comment.

```
#define TPM_NUM_DELEGATE_TABLE_ENTRY_MIN 2

typedef struct tdTPM_DELEGATE_TABLE{
   TPM_DELEGATE_TABLE_ROW delRow[TPM_NUM_DELEGATE_TABLE_ENTRY_MIN];
} TPM_DELEGATE_TABLE;
```

Parameters

Type	Name	Description
TPM_DELEGATE_TABLE_ROW	delRow	The array of delegations

20.12 TPM_DELEGATE_SENSITIVE

Start of informative comment:

The TPM_DELEGATE_SENSITIVE structure is the area of a delegate blob that contains sensitive information.

This structure is informative as the TPM vendor can include additional information. This structure is under complete control of the TPM and is never seen by any entity other then internal TPM processes.

End of informative comment.

```
typedef struct tdTPM_DELEGATE_SENSITIVE {
   TPM_STRUCTURE_TAG tag;
   TPM_SECRET authValue;
} TPM_DELEGATE_SENSITIVE;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This MUST be TPM_DELEGATE_SENSITIVE
TPM_SECRET	authValue	Authorization value

20.13 TPM_DELEGATE_OWNER_BLOB

Start of informative comment:

This data structure contains all the information necessary to externally store a set of owner delegation rights that can subsequently be loaded or used by this TPM.

The encryption mechanism for the sensitive area is a TPM choice. The TPM may use asymmetric encryption and the SRK for the key. The TPM may use symmetric encryption and a secret key known only to the TPM.

End of informative comment.

```
typedef struct tdTPM_DELEGATE_OWNER_BLOB{
   TPM_STRUCTURE_TAG tag;
   TPM_DELEGATE_PUBLIC pub;
   TPM_DIGEST integrityDigest;
   UINT32 additionalSize;
   [size_is(additionalSize)] BYTE* additionalArea;
   UINT32 sensitiveSize;
   [size_is(sensitiveSize)] BYTE* sensitiveArea;
} TPM_DELEGATE_OWNER_BLOB;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This MUST be TPM_TAG_DELG_OWNER_BLOB
TPM_DELEGATE_PUBLIC	pub	The public information for this blob
TPM_DIGEST	integrityDigest	The HMAC to guarantee the integrity of the entire structure
UINT32	additionalSize	The size of the integrity area
BYTE	additionalArea	An area that the TPM can add to the blob which MUST NOT contain any sensitive information. This would include any IV material for symmetric encryption
UINT32	sensitiveSize	The size of the sensitive area
BYTE	sensitiveArea	The area that contains the encrypted TPM_DELEGATE_SENSITIVE

20.14 TPM_DELEGATE_KEY_BLOB

Start of informative comment:

A structure identical to TPM_DELEGATE_OWNER_BLOB but which stores delegation information for user keys. As compared to TPM_DELEGATE_OWNER_BLOB, it adds a hash of the corresponding public key value to the public information.

End of informative comment.

```
typedef struct tdTPM_DELEGATE_KEY_BLOB{
   TPM_STRUCTURE_TAG tag;
   TPM_DELEGATE_PUBLIC pub;
   TPM_DIGEST integrityDigest;
   TPM_DIGEST pubKeyDigest;
   UINT32 additionalSize;
   [size_is(additionalSize)] BYTE* additionalArea;
   UINT32 sensitiveSize;
   [size_is(sensitiveSize)] BYTE* sensitiveArea;
} TPM_DELEGATE_KEY_BLOB;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This MUST be TPM_TAG_DELG_KEY_BLOB
TPM_DELEGATE_PUBLIC	pub	The public information for this blob
TPM_DIGEST	integrityDigest	The HMAC to guarantee the integrity of the entire structure
TPM_DIGEST	pubKeyDigest	The digest, that uniquely identifies the key for which this usage delegation applies. This is a hash of the TPM_STORE_PUBKEY structure.
UINT32	additionalSize	The size of the integrity area
BYTE	additionalArea	An area that the TPM can add to the blob which MUST NOT contain any sensitive information. This would include any IV material for symmetic encryption
UINT32	sensitiveSize	The size of the sensitive area
BYTE	sensitiveArea	The area that contains the encrypted TPM_DELEGATE_SENSITIVE

20.15 TPM_FAMILY_OPERATION Values

Start of informative comment:

These are the opFlag values used by TPM_Delegate_Manage.

End of informative comment.

Value	Capability Name	Comments
0x00000001	TPM_FAMILY_CREATE	Create a new family
0x00000002	TPM_FAMILY_ENABLE	Set or reset the enable flag for this family.
0x00000003	TPM_FAMILY_ADMIN	Prevent administration of this family..
0x00000004	TPM_FAMILY_INVALIDATE	Invalidate a specific family row.

21. Capability areas

21.1 TPM_CAPABILITY_AREA

Start of informative comment:

To identify a capability to be queried.

End of informative comment.

TPM_CAPABILITY_AREA Values

Value	Capability Name	Comments
0x00000001	TPM_CAP_ORD	Queries whether a command is supported.
0x00000002	TPM_CAP_ALG	Queries whether an algorithm is supported.
0x00000003	TPM_CAP_PID	Queries whether a protocol is supported.
0x00000004	TPM_CAP_FLAG	Queries whether a flag is on or off.
0x00000005	TPM_CAP_PROPERTY	Determines a physical property of the TPM.
0x00000006	TPM_CAP_VERSION	Queries the current TPM version.
0x00000007	TPM_CAP_KEY_HANDLE	Obtains information about key handles
0x00000008	TPM_CAP_CHECK_LOADED	Obtains information about the ability to load a key
0x00000009	TPM_CAP_BIT_OWNER	Get a bit from the Owner managed area
0x0000000A	TPM_CAP_BIT_LOCAL	Get a bit from the LOCAL_MOD managed area
0x0000000B	TPM_CAP_DELEGATIONS	Get information about delegations
0x0000000C	TPM_CAP_KEY_STATUS	Get information relative to a key and it's owner evict status
0x0000000D	TPM_CAP_NV_LIST	Get the list of NV storage indexes.
0x0000000E	TPM_CAP_TABLE_ADMIN	Get the table admin status
0x0000000F	TPM_CAP_TABLE_ENABLE	Get the table admin status
0x00000010	TPM_CAP_MFR	Manufacturer specific
0x00000011	TPM_CAP_NV_INDEX	The NV index values
0x00000012	TPM_CAP_TRANS_ALG	Algorithms supported for transport sessions
0x00000013	TPM_CAP_GPIO_CHANNEL	Information on the GPIO channel
0x00000014	TPM_CAP_HANDLE	Information regarding all handles in the system
0x00000015	TPM_CAP_TRANS_ES	Encryption scheme supported by transport sessions

21.2 GetCapability subCap definitions

Start of informative comment:

When in use what a subcap is defining

End of informative comment.

TPM_CAPABILITY_AREA Values

Value	Capability Name	Comments
0x00000101	TPM_CAP_PROP_PCR	
0x00000102	TPM_CAP_PROP_DIR	
0x00000103	TPM_CAP_PROP_MANUFACTURER	
0x00000104	TPM_CAP_PROP_KEYS	
0x00000107	TPM_CAP_MIN_COUNTER	
0x00000108	TPM_CAP_FLAG_PERMANENT	
0x00000109	TPM_CAP_FLAG_STCLEAR	
0x0000010A	TPM_CAP_PROP_AUTHSESS	
0x0000010B	TPM_CAP_PROP_TRANSESS	
0x0000010C	TPM_CAP_PROP_COUNTERS	
0x0000010D	TPM_CAP_PROP_MAX_AUTHSESS	
0x0000010E	TPM_CAP_PROP_MAX_TRANSESS	
0x0000010F	TPM_CAP_PROP_MAX_COUNTERS	
0x00000110	TPM_CAP_PROP_MAX_KEYS	
0x00000111	TPM_CAP_PROP_OWNER	
0x00000112	TPM_CAP_PROP_CONTEXT	
0x00000113	TPM_CAP_PROP_MAX_CONTEXT	
0x00000114	TPM_CAP_PROP_FAMILYROWS	
0x00000115	TPM_CAP_PROP_TIS	
0x00000116	TPM_CAP_PROP_STARTUP_EFFECT	
0x00000117	TPM_CAP_PROP_DELEGATE_ENTRIES	
0x00000118	TPM_CAP_PROP_NV_MAXBUF	
0x00000119	TPM_CAP_PROP_DAA_MAX	
0x0000011A	TPM_CAP_PROP_GLOBALLOCK	
0x0000011B	TPM_CAP_PROP_CONTEXT_DIST	
0x0000011C	TPM_CAP_PROP_DAA_INTERRUPT	
0X0000011D	TPM_CAP_FLAG_STANY	
0x0000011E	TPM_CAP_GPIO_CHANNEL	
0x0000011F	TPM_CAP_PROP_CMK_RESTRICTION	

1. The permitted values of TPM_CAP_PROP_MANUFACTURER and their meaning SHALL be defined in platform specific TPM specifications.

2. Ordering of TPM_CAP_FLAG_PERSISTENT
   a. The bits of the structure are marshaled in the following manner
   b. Bit-N of the TPM_PERMANENT_FLAGS structure is the Nth bit after the opening bracket in the definition of TPM_PERMANENT_FLAGS in the version of the specification indicated by the parameter “version”. The bit immediately after the opening bracket is the 0th bit.
   c. Bit-N of non_volatile_flags corresponds to the Nth bit in TPM_PERMANENT_FLAGS, and the lsb of non_volatile_flags corresponds to bit0 of TPM_PERMANENT_FLAGS
3. Ordering of TPM_CAP_FLAG_VOLATILE
   a. Bit-N of the TPM_VOLATILE_FLAGS structure is the Nth bit after the opening bracket in the definition of TPM_VOLATILE_FLAGS in the version of the specification indicated by the parameter “version”. The bit immediately after the opening bracket is the 0th bit.
   b. Bit-N of volatile_flags corresponds to the Nth bit in TPM_VOLATILE_FLAGS, and the lsb of volatile_flags corresponds to bit0 of TPM_VOLATILE_FLAGS
4. TIS Timeout values
   a. The short time is for commands that do not perform any RSA operations
   b. The medium time is for commands that perform RSA encrypt or decrypt operations
   c. The long time is for commands that perform RSA key generation

22. DAA Structures

All byte and bit areas are byte arrays treated as large integers

22.1 Size definitions

```
#define DAA_SIZE_r0      43 (Bytes)
#define DAA_SIZE_r1      43 (Bytes)
#define DAA_SIZE_r2     128 (Bytes)
#define DAA_SIZE_r3     158 (Bytes)
#define DAA_SIZE_r4     219 (Bytes)
#define DAA_SIZE_NT      20 (Bytes)
#define DAA_SIZE_u11    138 (Bytes)
#define DAA_SIZE_u2     128 (Bytes)
#define DAA_SIZE_u3     189 (Bytes)
#define DAA_SIZE_NE     256 (Bytes)
#define DAA_SIZE_w      256 (Bytes)
#define DAA_SIZE_v0     128 (Bytes)
#define DAA_SIZE_v1     190 (Bytes)
#define DAA_SIZE_count    1 (Byte)
#define DAA_SIZE_issuerModulus      256 (Bytes)
```

22.2 Constant definitions

```
#define DAA_power0      104
#define DAA_power1      1024
```

22.3 Error codes

Name	Value	Description
TPM_DAA_INPUT_DATA0		The consistency check on DAA parameter inputData0 has failed.
TPM_DAA_INPUT_DATA1		The consistency check on DAA parameter inputData1 has failed.
TPM_DAA_ISSUER_SETTINGS		The consistency check on DAA_issuerSettings has failed.
TPM_DAA_TPM_SETTINGS		The consistency check on DAA_tpmSpecific has failed.
TPM_DAA_STAGE		The atomic process indicated by the submitted DAA command is not the expected atomic process.
TPM_DAA_ISSUER_VALIDITY		The issuer's validity check has detected an inconsistency
TPM_DAA_WRONG_W		The consistency check on w has failed.

22.4 Digest Redefinitions

TPM_DIGEST	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_DIGEST	TPM_DAA_CONTEXT_SEED	This SHALL be a random value

22.5 Additions to Permanent Data

TPM_DAA_TPM_SEED	tpmDAASeed	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

22.6 Additions to Structure Tags

TPM_TAG_DAA_ISSUER	0X00000028	TPM_DAA_ISSUER
TPM_TAG_TPM_DAA_TPM	0X00000029	TPM_DAA_TPM
TPM_TAG_TPM_DAA_CONTEXT	0X0000002A	TPM_DAA_CONTEXT
TPM_TAG_TPM_DAA_BLOB	0x0000002C	TPM_DAA_BLOB
TPM_TAG_TPM_DAA_SENSITIVE	0X0000002D	TPM_DAA_SENSITIVE

22.7 TPM_DAA_ISSUER

Start of informative comment:

This structure is the abstract representation of non-secret settings controlling a DAA context. The structure is required when loading public DAA data into a TPM.

TPM_DAA_ISSUER parameters are normally held outside the TPM as plain text data, and loaded into a TPM when a DAA session is required. A TPM_DAA_ISSUER structure contains no integrity check: the TPM_DAA_ISSUER structure at time of JOIN is indirectly verified by the issuer during the JOIN process, and a digest of the verified TPM_DAA_ISSUER structure is held inside the TPM_DAA_TPM structure created by the JOIN process.

Parameters DAA_digest_X are digests of public DAA_generic_X parameters, and used to verify that the correct value of DAA_generic_X has been loaded. DAA_generic_q is stored in its native form to reduce command complexity.

End of informative comment.

Definition

```
typedef struct tdTPM_DAA_ISSUER {
   TPM_STRUCTURE_TAG    tag;
   TPM_DIGEST     DAA_digest_R0;
   TPM_DIGEST     DAA_digest_R1;
   TPM_DIGEST     DAA_digest_S0;
   TPM_DIGEST     DAA_digest_S1;
   TPM_DIGEST     DAA_digest_n;
   TPM_DIGEST     DAA_digest_n1;
   TPM_DIGEST     DAA_digest_gamma;
   BYTE[26]       DAA_generic_q;
} TPM_DAA_ISSUER;
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_TPM_DAA_ISSUER
TPM_DIGEST	DAA_digest_R0	A digest of the parameter “R0”, which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_R1	A digest of the parameter “R1”, which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_S0	A digest of the parameter “S0”, which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_S1	A digest of the parameter “S1”, which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_n	A digest of the parameter “n”, which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_n1	A digest of the parameter “n1”, which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_gamma	A digest of the parameter “gamma”, which is not secret and may be common to many TPMs.
BIT[]	DAA_ generic _q	The parameter q, which is not secret and may be common to many TPMs. Note that q is slightly larger than a digest, but is stored in its native form to simplify the TPM_DAA_join command. Otherwise, JOIN requires 3 input parameters.

22.8 TPM_DAA_TPM

Start of informative comment:

This structure is the abstract representation of TPM specific parameters used during a DAA context. TPM-specific DAA parameters may be stored outside the TPM, and hence this structure is needed to save private DAA data from a TPM, or load private DAA data into a TPM.

If a TPM_DAA_TPM structure is stored outside the TPM, it is stored in a confidential format that can be interpreted only by the TPM created it. This is to ensure that secret parameters are rendered confidential, and that both secret and non-secret data in TPM_DAA_TPM form a self-consistent set.

TPM_DAA_TPM includes a digest of the public DAA parameters that were used during creation of the TPM_DAA_TPM structure. This is needed to verify that a TPM_DAA_TPM is being used with the public DAA parameters used to create the TPM_DAA_TPM structure.

Parameters DAA_digest_v0 and DAA_digest_v1 are digests of public DAA_private_v0 and DAA_private_v1 parameters, and used to verify that the correct private parameters have been loaded.

Parameter DAA_count is stored in its native form, because it is smaller than a digest, and is required to enforce consistency.

End of informative comment.

Definition

```
typedef struct tdTPM_DAA_TPM {
   TPM_STRUCTURE_TAG tag;
   TPM_DIGEST     DAA_digestIssuer;
   TPM_DIGEST     DAA_digest_v0;
   TPM_DIGEST     DAA_digest_v1;
   TPM_DIGEST     DAA_rekey;
   UINT32         DAA_count;
} TPM_DAA_TPM;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_TPM_DAA_TPM
TPM_DIGEST	DAA_digestIssuer	A digest of a TPM_DAA_ISSUER structure that contains the parameters used to generate this TPM_DAA_TPM structure.
TPM_DIGEST	DAA_digest_v0	A digest of the parameter “v0”, which is secret and specific to this TPM. “v0” is generated during a JOIN phase.
TPM_DIGEST	DAA_digest_v1	A digest of the parameter “v1”, which is secret and specific to this TPM. “v1” is generated during a JOIN phase.
TPM_DIGEST	DAA_rekey	A digest related to the rekeying process, which is not secret but is specific to this TPM, and must be consistent across JOIN/SIGN sessions. “rekey” is generated during a JOIN phase.
UINT32	DAA_count	The parameter “count”, which is not secret but must be consistent across JOIN/SIGN sessions. “count” is an input to the TPM from the host system.

22.9 TPM_DAA_CONTEXT

Start of informative comment:

TPM_DAA_CONTEXT structure is created and used inside a TPM, and never leaves the TPM. This entire section is informative as the TPM does not expose this structure.

TPM_DAA_CONTEXT includes a digest of the public and private DAA parameters that were used during creation of the TPM_DAA_CONTEXT structure. This is needed to verify that a TPM_DAA_CONTEXT is being used with the public and private DAA parameters used to create the TPM_DAA_CONTEXT structure.

End of informative comment.

Definition

```
typedef struct tdTPM_DAA_CONTEXT {
   TPM_STRUCTURE_TAG    tag;
   TPM_DIGEST           DAA_digestContext
   TPM_DIGEST           DAA_digest;
   TPM_DAA_CONTEXT_SEED DAA_contextSeed;
   BYTE[256]            DAA_scratch;
   BYTE                 DAA_stage;
} TPM_DAA_CONTEXT;
```

Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_TPM_DAA_CONTEXT
TPM_DIGEST	DAA_digestContext	A digest of parameters used to generate this structure. The parameters vary, depending on whether the session is a JOIN session or a SIGN session.
TPM_DIGEST	DAA_digest	A running digest of certain parameters generated during DAA computation; operationally the same as a PCR (which holds a running digest of integrity metrics).
TPM_DAA_CONTEXT_SEED	DAA_contextSeed	The seed used to generate other DAA session parameters
BYTE[]	DAA_scratch	Memory used to hold different parameters at different times of DAA computation, but only one parameter at a time. The maximum size of this field is 256 bytes
BYTE	DAA_stage	A counter, indicating the stage of DAA computation that was most recently completed. The value of the counter is zero if the TPM currently contains no DAA context. When set to zero (0) the TPM MUST clear all other fields in this structure. The TPM MUST set DAA_stage to 0 on TPM_Startup(ANY)

22.10 TPM_DAA_JOINDATA

Start of informative comment:

This structure is the abstract representation of data that exists only during a specific JOIN session.

End of informative comment.

Definition

```
typedef struct tdTPM_DAA_JOINDATA {
   BYTE[128]      DAA_join_u0;
   BYTE[128]      DAA_join_u1;
   TPM_DIGEST     DAA_digest_n0;
} TPM_DAA_JOINDATA;
```

Parameters

Type	Name	Description
BYTE[]	DAA_join_u0	A TPM-specific secret “u0”, used during the JOIN phase, and discarded afterwards.
BIT[]	DAA_join_u1	A TPM-specific secret “u1”, used during the JOIN phase, and discarded afterwards.
TPM_DIGEST	DAA_digest_n0	A digest of the parameter “n0”, which is an RSA public key with exponent 2^16 +1

22.11 TPM_STANY_DATA Additions

Informative comment

This shows that the volatile data areas are added to the TPM_STANY_DATA structure

End of informative comment.

IDL Definition

```
typedef struct tdTPM_VOLATILE_DATA{
   TPM_DAA_ISSUER       DAA_issuerSettings;
   TPM_DAA_TPM          DAA_tpmSpecific;
   TPM_DAA_CONTEXT      DAA_session;
   TPM_DAA_JOINDATA     DAA_joinSession
}TPM_VOLATILE_DATA;
```

Types of Volatile Data

Type	Name	Description
TPM_DAA_ISSUER	DAA_issuerSettings	A set of DAA issuer parameters controlling a DAA session.
TPM_DAA_TPM	DAA_tpmSpecific	A set of DAA parameters associated with a specific TPM.
TPM_DAA_CONTEXT	DAA_session	A set of DAA parameters associated with a DAA session.
TPM_DAA_JOIN	DAA_joinSession	A set of DAA parameters used only during the JOIN phase of a DAA session, and generated by the TPM.

22.12 TPM_DAA_BLOB

Informative comment

The structure passed during the join process

End of informative comment.

Definition

```
typedef struct tdTPM_DAA_BLOB {
   TPM_STRUCTURE_TAG tag;
   TPM_RESOURCE_TYPE resourceType;
   BYTE[16] label;
   TPM_DIGEST blobIntegrity;
   UINT32 additionalSize;
   [size_is(additionalSize)] BYTE* additionalData;
   UINT32 sensitiveSize;
   [size_is(sensitiveSize)] BYTE* sensitiveData;
}TPM_DAA_BLOB;
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_BLOB
TPM_RESOURCE_TYPE	resourceType	The resource type: enc(DAA_tpmSpecific) or enc(v0) or enc(v1)
BYTE[16]	label	Label for identification of the blob. Free format area.
TPM_DIGEST	blobIntegrity	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 shielded locations. This should include any IV for symmetric encryption
UINT32	sensitiveSize	The size of sensitiveData
BYTE	sensitiveData	A TPM_DAA_SENSITIVE structure

22.13 TPM_DAA_SENSITIVE

Informative comment

The encrypted area for the DAA parameters

End of informative comment.

Definition

```
typedef struct tdTPM_DAA_SENSITIVE {
   TPM_STRUCTURE_TAG tag;
   UINT32 internalSize;
   [size_is(internalSize)] BYTE* internalData;
}TPM_DAA_SENSITIVE;
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_SENSITIVE
UINT32	internalSize	The size of the internalData area
BYTE	internalData	DAA_tpmSpecific or DAA_private_v0 or DAA_private_v1

23. GPIO structures

23.1 TPM_GPIO_BUS

Informative comment

The type(s) of data transfer channels that are supported by a TPM. The TPM is not required to support all of the listed busses. The platform specific specification will indicate the busses the TPM must support.

End of informative comment.

Command modes

Name	Value	Description
TPM_GPIO_SINGLE	0x00000001	A single pin bus
TPM_GPIO_SMBUS	0x00000002	The channel uses the SMBus block read/write protocol without PEP. The TPM acts as bus master and generates an SCL clock with a minimum clock rate of 10KHz. The first byte of the address is the slave address and the second byte is the command byte.
TPM_GPIO_SMBUS_ARP	0x00000003	The channel uses SMBus block write as GPIO_SMBUS. The output parameter readData is filled in with the byte number in the block write after which the slave returned a NACK or 0xFF if all bytes were ACK'd.

23.2 TPM_GPIO_ATTRIBUTES

Informative comment

The attribute flags for the channel

End of informative comment.

TPM_FAMILY_FLAGS bit settings

Bit Number	Bit Name	Comments
31:5	Reserved MUST be 0	
5	TPM_GPIO_ATTR_REDIR_KEY	The channel can be written only through the redirected key whose hash is specified in the TPM_GPIO_CHANNEL structure.
4	TPM_GPIO_ATTR_REDIR	The channel can be written only through a redirected key
3	TPM_GPIO_ATTR_WRITE	The channel can be written
2	TPM_GPIO_ATTR_READ	The channel can be read
1	TPM_GPIO_ATTR_PP	Physical Presence required to use channel
0	TPM_GPIO_ATTR_AUTH	Knowledge of the authorization value is required to use this channel with the TPM_GPIO_ReadWrite command.

23.3 TPM_GPIO_CHANNEL

Informative comment

Public information that defines information about the types of IO permitted on the channel identified by the TPM-assigned logical channel number held in the channelNumber field of this structure.

Bus information is duplicated in this structure to ease the job of software using the channel. This allows a single collection point for information relative to the channel instantiation.

End of informative comment.

Definition

```
typedef struct tdTPM_GPIO_CHANNEL {
   TPM_STRUCTURE_TAG tag;
   TPM_PLATFORM_SPECIFIC ps;
   UINT16 channelNumber;
   TPM_GPIO_ATTRIBUTES attr;
   TPM_GPIO_BUS busInfo;
   UINT32 sizeOfAddress;
   BYTE address [ ];
   UINT32 sizeOfPubKey;
   TPM_DIGEST pubKey;
   UINT32 sizeOfPcrInfo;
   TPM_PCR_INFO_SHORT pcrInfo;
}TPM_GPIO_CHANNEL;
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CONFIG_CHANNEL
TPM_PLATFORM_SPECIFIC	ps	The platform specific enumeration which indicates where the channel definition comes from
UINT16	channelNumber	The channel number being used
TPM_GPIO_ATTRIBUTES	attr	The attributes for this instantianation of the channel. The attributes MAY be a subset of the channel definition
TPM_GPIO_BUS	busInfo	The bus type.
UINT32	addressSize	The size of address in bytes, for an unstructured indicator channel this must be zero
BYTE	address	The structured bus address this channel is currently connected to
UINT32	pubKeySize	0 if the channel is not connected to a redirected key, 20 otherwise
TPM_DIGEST	pubKey	The public key to which this channel must be attached; ignored by TPM_GPIO_ReadWrite and ignored if channel RedirKey attribute is FALSE.
UINT32	pcrInfoSize	0 if the channel authorization data is not tied to PCRs or Locality
TPM_PCR_INFO_SHORT	pcrInfo	PCR values and/or Locality information that is necessary to access this channel. If PCRs not required, then pcrInfo->pcrSelect is 0. If Localities are not required then localityAtRelease is set to 0x1F.

23.4 TPM_GPIO_AUTHORIZE

Informative comment

The owner uses TPM_GPIO_AuthChannel command to build structures of this type to authorize later use of the specified IO channel. Because the structure includes an HMAC of the entire element using tpmProof as the key, it cannot be used for any other TPM.

If the authorize attribute is FALSE for this IO channel, then additionalSize and sensitiveSize would be set to 0.

End of informative comment.

Definition

```
typedef struct tdTPM_GPIO_AUTHORIZE {
   TPM_STRUCTURE_TAG tag;
   TPM_GPIO_CHANNEL channel;
   TPM_DIGEST blobIntegrity;
   UINT32 additionalSize;
   [size_is(additionalSize)] BYTE* additionalData;
   UINT32 sensitiveSize;
   [size_is(sensitiveSize)] BYTE* sensitiveData;
}TPM_GPIO_AUTHORIZE;
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_AUTHORIZE
TPM_GPIO_CHANNEL	channel	The channel being authorized by this block
TPM_DIGEST	blobIntegrity	The integrity of the entire blob including the sensitive area. This is a HMAC calculation with the entire structure (including sensitiveData before it is encrypted) using tpmProof as the secret
UINT32	additionalSize	The size of additionalData
BYTE	additionalData	Additional non-sensitive information necessary to load and verify the IO channel. This should include any IV for symmetric encryption
UINT32	sensitiveSize	The size of sensitiveData
BYTE	sensitiveData	An encrypted GPIO_SENSITIVE structure

23.5 TPM_GPIO_SENSITIVE

Informative comment

Secret information necessary to verify the authorization of the IO channel is included in this structure and encrypted before inclusion in the TPM_GPIO_CHANNEL structure.

End of informative comment.

Definition

```
typedef struct tdTPM_GPIO_SENSITIVE {
   TPM_STRUCTURE_TAG tag;
   TPM_DIGEST authData;
}TPM_GPIO_SENSITIVE;
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_SENSITIVE
TPM_DIGEST	authData	The authorization data for this channel.

24. Redirection

24.1 TPM_REDIR_COMMAND

Informative comment

The types of redirections

End of informative comment.

Command modes

Name	Value	Description
TPM_REDIR_GPIO	0x00000001	Redirect to a GPIO channel

25. Deprecated Structures

25.1 Persistent Flags

Start of Informative comment:

Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

End of informative comment.

```
typedef struct tdTPM_PERMANENT_FLAGS{
// deleted see version 1.1
} TPM_PERMANENT_FLAGS;
```

25.2 Volatile Flags

Start of Informative comment:

Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

End of informative comment.

```
typedef struct tdTPM_VOLATILE_FLAGS{
// see version 1.1
} TPM_VOLATILE_FLAGS;
```

25.3 TPM persistent data

Start of Informative comment:

Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

End of informative comment.

IDL Definition

```
typedef struct tdTPM_PERSISTENT_DATA{
// see version 1.1
}TPM_PERSISTENT_DATA;
```

25.4 TPM volatile data

Start of Informative comment:

Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

End of informative comment.

IDL Definition

```
typedef struct tdTPM_VOLATILE_DATA{
// see version 1.1
}TPM_VOLATILE_DATA;
```

25.5 TPM SV data

Start of Informative comment:

Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

End of informative comment.

IDL Definition

```
typedef struct tdTPM_SV_DATA{
// see version 1.1
}TPM_SV_DATA;
```

25.6 TPM_SYM_MODE

Start of informative comment:

This indicates the mode of a symmetric encryption. Mode is Electronic CookBook (ECB) or some other such mechanism.

End of informative comment.

TPM_SYM_MODE values

Value	Name	Description
0x00000001	TPM_SYM_MODE_ECB	The electronic cookbook mode (this requires no IV)
0x00000002	TPM_SYM_MODE_CBC	Cipher block chaining mode
0X00000003	TPM_SYM_MODE_CFB	Cipher feedback mode

Description

The TPM MAY support any of the symmetric encryption modes

End of document