TRUSTED® COMPUTING GROUP

GUIDANCE

TCG FIPS 140-3 guidance for TPM 2.0

Version 1.0
Revision 1
January 30, 2024

Contact: admin@trustedcomputinggroup.org

Public Review

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, DOCUMENT OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this document and to the implementation of this document, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this document or any information herein.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on document licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
Version 1 Revision 1	January 30, 2024	First revision for public review based on rev0.32 internal draft

CONTENTS

FIGURES

TABLES

1 SCOPE

This FIPS 140-3 guidance for TPM 2.0 is a supporting document for FIPS 140-3 evaluation of a TPM 2.0 product compliant with the TPM 2.0 library specification.

The intended audience for this document includes TPM manufacturers, FIPS Cryptographic Module Validation Program laboratories and FIPS evaluators.

This document describes implementation recommendations and some extensions to the TPM Library specification necessary for a successful FIPS 140-3 evaluation.

This guidance addresses requirements that are mandatory for FIPS 140-3 overall level 1.

For FIPS 140-2 evaluations, the recommendations are described in another guidance “TCG FIPS 140-2 Guidance for TPM 2.0” 3.

Although TPM 2.0 products achieved FIPS 140-2 certifications with overall level 2, FIPS 140-3 includes new mandatory level 2 requirements related to authentication that cannot be satisfied by TPM 2.0 library level 0 version 1.59. The complete impact of those new authentication requirements is unknown at the time of writing this guidance. The scope for the first release of this guidance is therefore limited to FIPS 140-3 level 1.

This guidance covers the functional command scope described in the TPM 2.0 library specification level 0 revision 1.59. However, TPM vendors may adapt this guidance for functional scopes defined in other TPM 2.0 Library revisions.

The scope of the algorithms considered in this guidance is the algorithms defined as mandatory in at least one of the following specifications:

- *TCG PC Client Specific Platform TPM Profile for TPM 2.0 Version 1.04 Revision 37, February 3, 2020*
- *TCG PC Client Specific Platform TPM Profile for TPM 2.0 Version 1.05 Revision 14, September 4, 2020.*

This guidance may not be suitable for all products for various reasons, such as:

- If an implementation issue is found in a specific product for a specific algorithm, the product may report that algorithm as non-approved even though it is defined in this guidance as approved.
- If FIPS 140-3 standards and publications that define approved algorithms are updated after the release of this guidance, TPM vendors will have to update their products to maintain compliance with the most recent FIPS 140-3 specifications in order to successfully complete product evaluation.

2 Terms and Definitions

This section is deliberately left empty.

3 References

1. TCG Trusted Platform Module Specification
   https://www.trustedcomputinggroup.org/tpm-library-specification/
2. TCG PC Client Platform TPM Profile (PTP) Specification
   https://trustedcomputinggroup.org/resource/pc-client-platform-tpm-profile-ptp-specification/
3. TCG FIPS 140-2 Guidance for TPM 2.0
   https://trustedcomputinggroup.org/resource/tcg-fips-140-2-guidance-for-tpm-2-0/
4. ISO/IEC 19790
   Information Technology – Security techniques – Security requirements for security modules – corrected version 2015-12-15
5. ISO/IEC 24759
   Information Technology – Security techniques – Test requirements for security modules – third edition 2017-03
6. SP800-140
   FIPS 140-3 Derived test requirements (DTR), March 2020
   https://csrc.nist.gov/pubs/sp/800/140/final
7. SP800-140A
   CMVP documentation requirements, March 2020
   https://csrc.nist.gov/pubs/sp/800/140/a/final
8. SP800-140Br1
   CMVP security policy requirements, November 2023
   https://csrc.nist.gov/pubs/sp/800/140/b/r1/final
9. SP800-140C rev2
   CMVP approved security functions, August 17, 2023
   https://csrc.nist.gov/pubs/sp/800/140/c/r2/final
10. SP800-140D rev2
    CMVP approved sensitive security parameters generation and establishment methods, August 17, 2023
    https://csrc.nist.gov/pubs/sp/800/140/d/r2/final
11. FIPS 140-3 Implementation Guidance
    Implementation guidance for FIPS 140-3 and the Cryptographic Module Validation Program, November 22, 2023
    https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

4 FIPS 140-3 references

4.1 FIPS 140-3 documentation

The versions of the FIPS documents that are reflected in this TCG FIPS 140-3 guidance are listed in section 3 References.

This TCG FIPS 140-3 guidance might be updated to reflect changes in FIPS 140-3 documentation. The reader is warned that discrepancies may occur when the references of the documents applicable for FIPS 140-3 evaluation are different to the references provided in this TCG FIPS 140-3 guidance.

4.2 FIPS 140-3 acronyms

Table 1 lists the acronyms used in FIPS terminology and the corresponding algorithms supported by TPM 2.0 specifications. This table makes this TCG FIPS 140-3 guidance easier to understand by readers without FIPS expertise.

Table 1 — FIPS acronyms

FIPS terminology	Description
APT	Adaptive Proportion Test
CKG	Cryptographic Key Generation
DRBG	Deterministic Random Bit Generator
EDC	Error Detection Code
ENT	Entropy source
KAS	Key Agreement Scheme
KAT	Known Answer Test
KBKDF	Key-Based Key Derivation Function
KDA	Key Derivation Algorithm
KTS	Key Transport Scheme
PCT	Pair-wise Consistency Test
RCT	Repetition Count Test
SSC	Shared Secret Calculation
SHS	Secure Hash Standard

5 Algorithms

5.1 FIPS Approved cryptographic algorithms and security functions

The FIPS approved algorithms listed below are defined in

- SP800-140C rev2 and
- SP800-140D rev2

5.2 Approved algorithms

Table 2 —Approved algorithms

Algorithm	Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use/ Security Function Implementation	TCG algorithm reference
RSA	FIPS 186-4	RSASSA-PSS with SHA-256, SHA-384 and DRBG RSASSA-PKCS-v1.5 with SHA-256, SHA-384	2048 bits, 3072 bits	RSA signature generation	TPM_ALG_RSASSA TPM_ALG_RSAPSS[1] TPM_ALG_RSA
ECDSA	FIPS 186-4	SHA-256, SHA-384 DRBG	P-256, P-384	ECC signature generation	TPM_ALG_ECDSA TPM_ALG_ECC
HMAC	FIPS 198-1	SHA-1, SHA-256, SHA-384	160 bits, 256 bits, 384 bits	HMAC signature generation	TPM_ALG_HMAC
RSA	FIPS 186-4	RSASSA-PSS with SHA-1, SHA-256, SHA-384 RSASSA-PKCS-v1.5 with SHA-1, SHA-256, SHA-384	1024 bits, 2048 bits, 3072 bits	RSA signature verification	TPM_ALG_RSASSA TPM_ALG_RSAPSS TPM_ALG_RSA
ECDSA	FIPS 186-4	SHA-1, SHA-256, SHA-384	P-256, P-384	ECC signature verification	TPM_ALG_ECDSA TPM_ALG_ECC
HMAC	FIPS 198-1	SHA-1, SHA-256, SHA-384	160 bits, 256 bits, 384 bits	HMAC signature verification	TPM_ALG_HMAC
KTS RSA	SP800-56B Rev 2	KTS-OAEP-basic	2048 bits, 3072 bits	RSA key transport	TPM_ALG_OAEP TPM_ALG_RSA
KAS	SP 800-56A Rev3 SP 800-56C Rev1	-	P-256,P-384	ECC key agreement	TPM_ALG_ECDH TPM_ALG_ECC TPM_ALG_KDF1_SP800_56A
AES	FIPS 197 SP 800-38A	CFB128	128 bits, 256 bits	AES encryption/decryption	TPM_ALG_AES TPM_ALG_CFB
RSA	FIPS 186-4	-	2048 bits, 3072 bits	RSA key generation	TPM_ALG_RSA
ECDSA ECDH	FIPS 186-4	-	P-256,P-384	ECC key generation	TPM_ALG_ECC

[1] For TPM_ALG_RSAPSS signature scheme, the salt must be generated with an approved DRBG algorithm.

KBKDF	SP800-108	CTR	160 bits, 256 bits, 384 bits	Symmetric key derivation (KBKDF)	TPM_ALG_KDF1_SP800_108
KDA	SP 800-56C Rev1	SHA-1, SHA-256, SHA-384	160 bits, 256 bits, 384 bits	Symmetric key derivation (KDA)	TPM_ALG_KDF1_SP800_56A
KTS (AES + HMAC)	SP800-38F	CFB	128 bits, 256 bits	Symmetric key (un)wrapping	TPM_ALG_AES TPM_ALG_CFB TPM_ALG_HMAC
SHS	FIPS 180-4	SHA-1, SHA-256, SHA-384	160 bits, 256 bits, 384 bits	Message digest	TPM_ALG_SHA1 TPM_ALG_SHA256 TPM_ALG_SHA384
DRBG	SP800-90A, Rev1	Vendor dependent	-	Deterministic random bit generation	No reference
ENT	SP800-90B	-	-	DRBG seeding	No reference
KAS-SSC	SP 800-56A Rev3	-	P-256, P-384	ECC Shared Secret Calculation	TPM_ALG_ECDH TPM_ALG_ECC

5.3 Approved Vendor-Affirmed algorithms

Table 3 — Approved Vendor-affirmed algorithms

Algorithm	Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use/ Security Function Implementation	TCG algorithm reference
CKG	SP800-133, Rev2 (per FIPS 140-3 IG D.H)	Vendor dependent	160 bits, 256 bits, 384 bits	HMAC key generation	TPM_ALG_HMAC
			128 bits, 256 bits	AES key generation	TPM_ALG_AES

5.4 Non-approved algorithms allowed in the approved mode of operation

The TPM does not provide any non-approved algorithms that are allowed in the approved mode of operation.

5.5 Non-approved algorithms allowed in the approved mode of operation with no security claimed

Table 4 — Non-approved algorithms allowed in the approved mode of operation with no security claimed

Algorithm	Caveat	Use
XOR	Considered as obfuscation of input or output data (referenced as TPM_ALG_XOR in TCG TPM2.0 library specification).	Obfuscation or de-obfuscation of the first parameter of a command/response in an encryption or decryption session (§21.2 in TCG TPM2.0 Part1) Obfuscation or de-obfuscation of data with a keyed hash object

5.6 Non-approved algorithms not allowed in the approved mode of operation

Table 5 — Non-approved algorithms not allowed in the approved mode of operation

Algorithm	Use/ Function	TCG algorithm (or curve) reference
RSA	RSA key generation with RSA key length equal to 1024 bits	TPM_ALG_RSA
KTS RSA	RSA Key Transport with RSA key length equal to 1024 bits	TPM_ALG_RSA TPM_ALG_OAEP
RSA	RSA signature generation with RSA key length equal to 1024 bits	TPM_ALG_RSASSA TPM_ALG_RSAPSS TPM_ALG_RSA
ECDSA	ECC signature generation with SHA-1	TPM_ALG_ECDSA TPM_ALG_ECC
RSA	RSA signature generation with SHA-1	TPM_ALG_RSASSA TPM_ALG_RSAPSS TPM_ALG_RSA
KBKDF	Asymmetric ECC key derivation	TPM_ALG_KDF1_SP800-108 TPM_ALG_ECC
ECDSA	ECC Signature generation with Derived Asymmetric ECC Keys	TPM_ALG_ECDSA TPM_ALG_ECC
ECDSA	ECC Signature verification with Derived Asymmetric ECC Keys	TPM_ALG_ECDSA TPM_ALG_ECC
KAS	ECC key agreement with Derived Asymmetric ECC Keys	TPM_ALG_ECC TPM_ALG_ECDH TPM_ALG_KDF1_SP800-56A
KTS RSA	RSA Key Transport with Non Approved Padding scheme RSAES-PKCS-v1.5 [2]	TPM_ALG_RSAES TPM_ALG_RSA
KTS RSA	RSA Key Transport with Non Approved Padding scheme NULL	TPM_ALG_NULL TPM_ALG_RSA

[2], RSAES-PKCS-v1.5 is listed as non-approved because SP800-131Ar2 classifies it as disallowed after 2023.

Algorithm	Use/ Function	TCG algorithm (or curve) reference
CKG [3]	HMAC key generation with Key Size < 112 bits	TPM_ALG_HMAC
HMAC	HMAC signature generation and verification with Key Size < 112 bits	TPM_ALG_HMAC
ECDAA	ECDAA signature generation	TPM_ALG_ECDAA
ECSCHNORR	ECSCHNORR signature generation	TPM_ALG_ECSCHNORR
ECSCHNORR	ECSCHNORR signature verification	TPM_ALG_ECSCHNORR
ECDSA ECDH	ECC Key generation with BN P-256 elliptic curve	TPM_ECC_BN_P256 TPM_ALG_ECC
ECDSA	ECC signature generation with BN P-256 elliptic curve	TPM_ALG_ECDSA TPM_ALG_ECC TPM_ECC_BN_P256
ECDSA	ECC signature verification with BN P-256 elliptic curve	TPM_ALG_ECDSA TPM_ALG_ECC TPM_ECC_BN_P256
KAS	ECC key agreement with BN P-256 elliptic curve	TPM_ALG_ECDH TPM_ALG_ECC TPM_ECC_BN_P256 TPM_ALG_KDF1_SP800-56A
KAS-SSC	ECC Shared Secret Calculation with BN-P-256 elliptic curve	TPM_ALG_ECDH TPM_ALG_ECC TPM_ECC_BN_P256
KAS-SSC	ECC Shared Secret Calculation with Derived Asymmetric ECC Key	TPM_ALG_ECDH TPM_ALG_ECC

[3] HMAC key generation and HMAC signature generation/verification with a key size less than 112 bits can be supported by a TPM compliant with the TPM library specification in specific use cases, supported by the commands TPM2_Create, TPM2_Createloaded and TPM2_CreatePrimary. For those commands, if the parameter sensitiveDataOrigin is set to 0 (CLEAR), an external user may provide a HMAC key of any length that is used during command processing for HMAC generation or verification.

6 Security Function implementations

The Security Function Implementations that must be described in a non-proprietary vendor security policy are described in Table 2 —Approved algorithms and Table 3 — Approved Vendor-affirmed algorithms in column "Use/Security function implementation".

The links between the Approved services and the Approved algorithms are provided via a reference to the Security Function Implementation entries.

A vendor may instantiate the same Security Function Implementation several times in the non-proprietary security policy if the product supports several implementations of the same Approved Algorithm. In that case, the approved algorithms table Table 2 must be updated to refer to the specific Security Functional Implementation used for each approved service.

7 FIPS 140-3 indicator

7.1 FIPS 140-3 requirements

For convenience, this section summarizes verbatim the indicator-related requirements in FIPS 140-3 that a TPM must satisfy.

7.1.1 ISO/IEC 19790

ISO/IEC 19790 section 7.2.4.2: [4]

All services shall [02.24] provide an indicator when the service utilizes an approved cryptographic algorithm, security function or process in an approved manner and those services or processes specified in 7.4.3.

7.1.2 ISO/IEC 24759

ISO/IEC 24759, section 6.2.4.2:[5]

AS02.24: (Specification — Levels 1, 2, 3, and 4)

All services shall provide an indicator when the service utilizes an approved cryptographic algorithm, security function or process in an approved manner and those services or processes specified in {ISO/IEC 19790:2012} 7.4.3.

Required Vendor Information

VE02.24.01: The vendor provided documentation shall specify the indicator for each service.

Required Test Procedures

TE02.24.01: The tester shall verify that the vendor provided documentation contains a description of the indicator when the service utilizes an approved cryptographic algorithm, security function or process in an approved manner.

7.1.3 FIPS 140-3 Implementation Guidance

7.1.3.1 IG 2.4.C

From IG 2.4.C:

The Security Policy shall provide a complete list of all approved and non-approved services along with details on each service and their respective indicators (if applicable). The security policy may be used to provide interpretation for the indicator(s) provided by the module, but the description in the security policy alone does not fulfill the requirement.

The definition of the indicator follows option 3 in the example scenarios described in IG 2.4.C:

- *“Shared indicator for multiple approved services"*

combined with the implementation option

- *“Use of a dedicate query function for the operator to determine whether the current security service in use is approved”*

An operator queries the indicator to assess whether the last successfully executed command provided an approved or non-approved security service.

[4] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.

[5] ©ISO. This material is reproduced from ISO/IEC 24759:2017 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.

7.2 FIPS indicator for TPM implementations

This guidance proposes the addition of the FIPS_140_3 and FIPS_140_3_INDICATOR bits to the TPMA_MODES field definition in the TPM Library as shown in Table 6.

Table 6 — Definition of (UINT32) TPMA_MODES Bits <Out>

Bit	Name	Definition
0	FIPS_140_2	**SET (1):** indicates that the TPM is designed to comply with all of the FIPS 140-2 requirements at Level 1 or higher.
1	FIPS_140_3	**SET (1):** indicates that the TPM is designed to comply with all of the FIPS 140-3 requirements at Level 1 or higher.
3:2	FIPS_140_3_INDICATOR	Indicates the category of the service provided by the last command successfully executed before TPM2_GetCapability (capability = TPM_CAP_TPM_PROPERTIES, property = TPM_PT_MODES) command. This attribute is only meaningful if FIPS_140_3 is SET.
31:4	Reserved	shall be zero

The FIPS_140_3_INDICATOR values are:

(00): the service belongs to category 2, 3 or 5 (non-security relevant services)

(01): the service belongs to category 1 (approved security services)

(10): the service belongs to category 4 (non-approved security services)

(11): reserved value

NOTE 2 The categories 1 to 5 are defined in FIPS 140-3 IG 2.4.C

TPM products are recommended to

- SET the FIPS_140_3 bit
- Report FIPS_140_3_INDICATOR according to this guidance

7.3 CSP generation aspects

The following sub-sections discuss various aspects of determining the indicator value when generating CSPs.

7.3.1 Independence of TPM object parameters

TPM objects (CSPs) may be created with parameters that restrict the usage of the CSPs to specific algorithms once created. For instance, signature keys may be defined to be exclusively used with a specific signature scheme whereas other signature keys may be defined to be used with any signature scheme. TPM object parameters define how a CSP may be used after creation. An object may be:

- always used with approved services, or
- always used with non-approved services, or
- used with approved or non-approved services depending on TPM command parameters.

Since those object parameters do not impact generation, load or import services, this FIPS 140-3 guidance recommends reporting the generation, load or import services as approved provided the key was generated/loaded/imported using approved algorithms, even if the TPM object parameters do not enforce that the CSP can be used exclusively with approved services. For instance, an SSP can be generated by a TPM with object parameters that do not ensure that the SSP can be used exclusively with approved services. This SSP generation service will be reported as approved by the TPM. In a second step, this SSP may be exported to another FIPS certified security module that imports only the SSP's value without the TPM object parameters. The SSP's value may be used in the destination security module with only approved services. In that case, the SSP is used only with approved services that include the generation reported as a FIPS approved service by the TPM.

7.3.2 CSP generated with approved services and used with non-approved services

The following TPM Object properties cause a TPM to report SSP generation as approved but the security services that use the SSP will be reported as non-approved:

- RSA or ECC objects with *sign* attribute SET and *hashAlg* set to TPM_ALG_SHA1
    - **RSA or ECC signature** service must be reported as non-approved
- RSA objects with decrypt attribute SET and *padding scheme* TPM_ALG_RSAES
    - **RSA Key transport** must be reported as non-approved

Note that to be regarded as approved, CSP generation must satisfy the requirements in section 10.2.2.2. If the CSP is tested using a non-approved algorithm when the CSP is created, CSP generation is marked as non-approved.

7.3.3 CSP exclusivity between approved and non-approved services

ISO/IEC19790, section 7.2.4.2 states[6]:

*CSPs **shall [02.22]** be exclusive between approved and non-approved services and modes of operation (e.g. not shared or accessed).*

Some of the TPM objects parameters enforce SSP exclusivity between approved and non-approved services. However, parameters that enforce CSP exclusivity for a security service will be checked only once that specific service is provided by a TPM. A service will be reported as non-approved if the TPM object/SSP parameters are not set properly to enforce exclusivity for the specific service being rendered.

The following TPM Object properties will cause TPM Object key generation to be reported as approved but the security services that use that object will be reported as non-approved:

- RSA or ECC objects with *sign* attribute SET and *scheme* TPM_ALG_NULL
    - **RSA or ECC signature** service will be reported as non-approved
        - ECC objects might be used with both ECDSA (approved) and ECSHNORR\ECDAA (non-approved)
        - RSA/ECC objects might be used with both SHA-256 hash algorithm (approved) and SHA-1 hash algorithm (non-approved)
- RSA objects with decrypt attribute SET and *scheme* TPM_ALG_NULL
    - **RSA Key transport** will be reported as non-approved
        - RSA objects might be used with both OAEP (approved) and RSAES (non-approved)

7.3.4 Object parameters leading to non-approved generation service

Some object parameters provided during SSP generation will cause the generation service to be reported as non-approved. In that case, as the SSP generation is non-approved, all the security services applicable to those SSPs will be also non-approved. From an implementation point of view, a TPM object itself may be flagged as non-approved to simplify the reporting of the FIPS indicator across all services.

[6] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.

In order to consistently report the approved/ non-approved status of both imported objects and generated objects, object parameters that are checked during object generation are also checked during object importation.

The parameters that cause non-approved SSP generation service are checked in the imported SSP. An import service is reported as non-approved when the imported SSP is one of the following:

- an RSA key with a key length equal to 1024 bits
- an ECC Key with BN P-256 elliptic curve
- an HMAC key with Key Size < 112 bits

7.4 TPM Commands categories

Table 7 below lists for each TPM command the category defined by IG 2.4.C for a FIPS 140-3 evaluation at level 1. TPM's cryptographic services for command authorization are not considered.

The column "PTP 1.05" includes information from *TCG PC Client Specific Platform TPM Profile for TPM 2.0 Version 1.05* stating whether the command support is mandatory or optional.

The CH numbers in the "Commands" column are the numbers of chapters in the TPM specification.

Table 7 — TPM Commands overview

Commands	PTP 1.05 classification mandatory or optional	IG 2.4.C category
Startup (CH9)		
TPM2_Startup	M	2, 3, 5
TPM2_Shutdown	M	2, 3, 5
Testing (CH10)		
TPM2_IncrementalSelfTest	M	1 OR 2,3,5
TPM2_SelfTest	M	1 OR 2,3,5
TPM2_GetTestResult	M	2,3,5
Session Commands (CH11)		
TPM2_StartAuthSession	M	1 or 4
TPM2_PolicyRestart	M	2, 3, 5
Object Commands (CH 12)		
TPM2_Create	M	1 or 4
TPM2_Load	M	1 or 4
TPM2_LoadExternal	M	1 or 4

Commands	PTP 1.05 classification mandatory or optional	IG 2.4.C category
TPM2_ReadPublic	M	2, 3, 5
TPM2_ActivateCredential	M	1 or 4
TPM2_MakeCredential	M	1 or 4
TPM2_Unseal	M	2, 3, 5
TPM2_ObjectChangeAuth	M	1
TPM2_CreateLoaded	M	1 or 4
Duplicate Commands (CH13)		
TPM2_Duplicate	M	1 or 4
TPM2_Rewrap	O	1 or 4
TPM2_Import	M	1 or 4
Asymmetric Primitives (CH14)		
TPM2_RSA_Encrypt	M	1 or 4
TPM2_RSA_Decrypt	M	1 or 4
TPM2_ECDH_KeyGen	M	1 or 4
TPM2_ECDH_ZGen	M	1 or 4
TPM2_ECC_Parameters	M	2, 3, 5
TPM2_Zgen_2Phase	O	4
Symmetric Primitives (CH15)		
TPM2_EncryptDecrypt	O	1
TPM2_EncryptDecrypt2	O	1
TPM2_Hash	M	1
TPM2_HMAC	O	1
TPM2_MAC	O	1 or 4
Random Number Generator (CH16)		
TPM2_GetRandom	M	1
TPM2_StirRandom	M	1
Hash/HMAC/Event Sequences (CH17)		

Commands	PTP 1.05 classification mandatory or optional	IG 2.4.C category
TPM2_HMAC_Start	M	1 or 4
TPM2_HashSequenceStart	M	1
TPM2_SequenceUpdate	M	1 or 4
TPM2_SequenceComplete	M	1 or 4
TPM2_EventSequenceComplete	M	1 or 4
Attestation Commands (CH18)		
TPM2_Certify	M	1 or 4
TPM2_CertifyCreation	M	1 or 4
TPM2_Quote	M	1 or 4
TPM2_GetSessionAuditDigest	M	1 or 4
TPM2_GetCommandAuditDigest	O	1 or 4
TPM2_GetTime	O	1 or 4
TPM2_CertifyX509		1 or 4
Anonymous Attestation (CH19)		
TPM2_Commit	M	4
TPM2_EC_Ephemeral	O	4
Signature Verification (CH20)		
TPM2_VerifySignature	M	1 or 4
TPM2_Sign	M	1 or 4
Command Audit (CH21)		
TPM2_SetCommandCodeAuditStatus	O	2, 3, 5
Integrity Collection (PCR) (CH22)		
TPM2_PCR_Extend	M	1
TPM2_PCR_Event	M	1
TPM2_PCR_Read	M	2, 3, 5
TPM2_PCR_Allocate	M	2, 3, 5
TPM2_PCR_SetAuthPolicy	O	2, 3, 5

Commands	PTP 1.05 classification mandatory or optional	IG 2.4.C category
TPM2_PCR_SetAuthValue	O	2, 3, 5
TPM2_PCR_Reset	M	2, 3, 5
Enhanced Authorization (EA) Commands (CH23)		
TPM2_PolicySigned	M	1 or 4
TPM2_PolicySecret	M	1 or 4
TPM2_PolicyTicket	O	1 or 4
TPM2_PolicyOR	M	1
TPM2_PolicyPCR	M	1
TPM2_PolicyLocality	M	1
TPM2_PolicyNV	M	1
TPM2_PolicyCounterTimer	M	1
TPM2_PolicyCommandCode	M	1
TPM2_PolicyPhysicalPresence	O	1
TPM2_PolicyCpHash	M	1
TPM2_PolicyNameHash	M	1
TPM2_PolicyDuplicationSelect	M	1
TPM2_PolicyAuthorize	M	1 or 4
TPM2_PolicyAuthValue	M	1
TPM2_PolicyPassword	M	1
TPM2_PolicyGetDigest	M	2,3,5
TPM2_PolicyNvWritten	M	1
TPM2_PolicyTemplate	M	1
TPM2_PolicyAuthorizeNV	M	1
Hierarchy Commands (CH24)		
TPM2_CreatePrimary	M	1 or 4
TPM2_HierarchyControl	M	2, 3, 5
TPM2_SetPrimaryPolicy	M	2, 3, 5

Commands	PTP 1.05 classification mandatory or optional	IG 2.4.C category
TPM2_ChangePPS	O	1
TPM2_ChangeEPS	O	1
TPM2_Clear	M	1
TPM2_ClearControl	M	2, 3, 5
TPM2_HierarchyChangeAuth	M	2, 3, 5
Dictionary Attack Functions (CH25)		
TPM2_DictionaryAttackLockReset	M	2, 3, 5
TPM2_DictionaryAttackParameters	M	2, 3, 5
Miscellaneous Management Functions (CH26)		
TPM2_PP_Commands	O	2, 3, 5
TPM2_SetAlgorithmSet	O	2, 3, 5
Field Upgrade (CH27)		
TPM2_FieldUpgradeStart	O	[7]
TPM2_FieldUpgradeData	O	
TPM2_FirmwareRead	O	
Context Management (CH28)		
TPM2_ContextSave	M	1
TPM2_ContextLoad	M	1
TPM2_FlushContext	M	2, 3, 5
TPM2_EvictControl	M	2, 3, 5
Clocks and Timers (CH29)		
TPM2_ReadClock	M	2, 3, 5
TPM2_ClockSet	M	2, 3, 5
TPM2_ClockRateAdjust	M	2, 3, 5
Capability Commands (CH30)		

[7] The implementation of firmware upgrade commands is proprietary. The choice of algorithms used to secure firmware upgrade is vendor specific and must be described in the vendor non-proprietary security policy.

Commands	PTP 1.05 classification mandatory or optional	IG 2.4.C category
TPM2_GetCapability	M	2, 3, 5
TPM2_TestParms	M	2, 3, 5
Non-volatile Storage (CH31)		
TPM2_NV_DefineSpace	M	2,3,5
TPM2_NV_UndefineSpace	M	2,3,5
TPM2_NV_UndefineSpaceSpecial	M	2,3,5
TPM2_NV_ReadPublic	M	1
TPM2_NV_Write	M	2,3,5
TPM2_NV_Increment	M	2,3,5
TPM2_NV_Extend	M	1
TPM2_NV_SetBits	M	2,3,5
TPM2_NV_WriteLock	M	2,3,5
TPM2_NV_GlobalWriteLock	O	2,3,5
TPM2_NV_Read	M	2,3,5
TPM2_NV_ReadLock	M	2,3,5
TPM2_NV_ChangeAuth	M	2,3,5
TPM2_NV_Certify	O	1 or 4
Attached Components Commands (CH32)		
TPM2_AC_GetCapability	O	2, 3, 5
TPM2_AC_Send		2, 3, 5
TPM2_Policy_AC_Send		2, 3, 5
Authenticated Countdown Timer (CH33)		
TPM2_ACT_SetTimeout	O	2, 3, 5

7.5 TPM Commands category 1 or 4

7.5.1 Service table template

SP800-140Br1 defines in sections 4.3 and 4.4 how approved services and non-approved services must be described in the vendor security policy. The description of both approved and non-approved services must include:

- A service name
- A description of the service purpose
- A list of approved or non-approved security functions used by the invocation of the service
- A list of SSPs associated with the service
- For level 2 evaluation, the list of roles and for each role, the list of the authorized services, the access rights to the SSPs and authentication method.

In this FIPS 140-3 guidance, the information provided for each approved and non-approved service is restricted to the following subset of the information required in SP800-140Br1:

- Service name: the name of the TPM command as described in the TCG Trusted Platform Module Specification
- Description of the service: a concise description of the TPM command's purpose
- Security functions: a list of security functions implemented during TPM command execution, either
  - Approved security functions listed in section 5.2 or
  - Non-Approved security functions listed in section 5.6

Table 8 describes the template of the TPM commands that provide approved services (category 1).

Table 8 — Approved service table template

Service	Description	Approved security functions
TPM Command	TPM command purpose	Link to approved security functions listed in section 5.2

Table 9 describes the template of the TPM commands that provide non-approved services (category 4).

Table 9 — Non-Approved service table template

Service	Description	Non-approved security functions
TPM Command	TPM command purpose	Link to non-approved security functions listed in section 5.6

In the following sections, TPM commands are grouped (in chapters) as in the TPM library specification part 3: commands. The chapter numbers of TPM specification part 3 are referenced in the section titles.

7.5.2 Chapter 10: Testing

Table 10 — Approved services for Testing commands

Service	Description	Approved Security Function
TPM2_IncrementalSelfTest	Perform SelfTest of selected algorithms	RSA signature generation ECC signature generation HMAC signature generation RSA signature verification ECC signature verification HMAC signature verification Symmetric key derivation (KDA) RSA key transport AES encryption/decryption Symmetric key derivation (KBKDF) Message digest Deterministic random bit generation DRBG seeding ECC Shared Secret Calculation
TPM2_SelfTest (fullTest = YES)	Perform SelfTest of all functions or only those that have not previously been tested	RSA signature generation ECC signature generation HMAC signature generation RSA signature verification ECC signature verification HMAC signature verification Symmetric key derivation (KDA) RSA key transport AES encryption/decryption Symmetric key derivation (KBKDF) Message digest Deterministic random bit generation DRBG seeding ECC Shared Secret Calculation

7.5.3 Chapter 11: Session Commands

Table 11 — Approved services for Session commands

Service	Description	Approved Security Function
TPM2_StartAuthSession	Start authorization session	RSA key transport ECC key agreement Deterministic random bit generation Symmetric key derivation (KBKDF)

Table 12 — Non-Approved services for Session commands

Service	Description	Non-Approved Security Function
TPM2_StartAuthSession	Start authorization session	RSA Key Transport with RSA key length equal to 1024 bits ECC key agreement with BN P-256 elliptic curve

7.5.4 Chapter 12: Object Commands

Table 13 — Approved services for Objects commands

Service	Description	Approved Security Function
TPM2_Create	Creation of an ordinary object	RSA key generation ECC key generation AES key generation HMAC key generation Deterministic random bit generation HMAC signature generation Message digest DRBG seeding Symmetric key derivation (KBKDF) Symmetric key (un)wrapping
TPM2_Load	Loading an ordinary object	Message digest Symmetric key derivation (KBKDF) Symmetric key (un)wrapping HMAC signature generation ECC Shared Secret Calculation
TPM2_LoadExternal	Loading an external object	Message digest ECC Shared Secret Calculation HMAC signature generation
TPM2_CreateLoaded	Creation and loading of an ordinary or a derived object	RSA key generation ECC key generation AES key generation HMAC key generation Deterministic random bit generation HMAC signature generation Message digest DRBG seeding Symmetric key derivation (KBKDF) Symmetric key (un)wrapping
TPM2_ObjectChangeAuth	Change the authorization secret of an object	Message digest Deterministic random bit generation Symmetric key derivation (KBKDF) Symmetric key (un)wrapping

TPM2_MakeCredential	encrypts an object credential	Deterministic random bit generation RSA key transport ECC key generation ECC key agreement Symmetric key derivation (KBKDF) Symmetric key (un)wrapping
TPM2_ActivateCredential	decrypts an object credential	RSA key transport ECC key agreement Symmetric key derivation (KBKDF) Symmetric key (un)wrapping

Table 14 — Non-Approved services for Objects commands

Service	Description	Non-Approved Security Function
TPM2_Create	Creation of an ordinary object	RSA key generation with RSA key length equal to 1024 bits ECC Key generation with BN P-256 elliptic curve HMAC key generation with Key Size < 112 bits
TPM2_Load	Loading an ordinary object	ECC Shared Secret Calculation with BN-P-256 elliptic curve ECC Shared Secret Calculation with Derived Asymmetric ECC Key SSP input check: RSA key loading with RSA key length equal to 1024 bits ECC Key loading with BN P-256 elliptic curve HMAC key loading with Key Size < 112 bits
TPM2_LoadExternal	Loading an external object	ECC Shared Secret Calculation with BN-P-256 elliptic curve ECC Shared Secret Calculation with Derived Asymmetric ECC Key[8] SSP input check: RSA key loading with RSA key length equal to 1024 bits ECC Key loading with BN P-256 elliptic curve HMAC key loading with Key Size < 112 bits
TPM2_CreateLoaded	Creation and loading of an ordinary or a derived object	RSA key generation with RSA key length equal to 1024 bits ECC Key generation with BN P-256 elliptic curve HMAC key generation with Key Size < 112 bits Asymmetric ECC key derivation
TPM2_MakeCredential	Encrypts an object credential	RSA Key Transport with RSA key length equal to 1024 bits ECC Key generation with BN P-256 elliptic curve ECC key agreement with BN P-256 elliptic curve
TPM2_ActivateCredential	Decrypts an object credential	RSA Key Transport with RSA key length equal to 1024 bits ECC key agreement with BN P-256 elliptic curve

[8] For the TPM2_LoadExternal command, both non-approved security functions are executed when loading an object with sensitive data under a parent object with BN-P256 elliptic curve or Derived Asymmetric ECC key. The shared secret calculation is linked to the Point Multiplication operation in the TPM reference simulator.

7.5.5 Chapter 13: Duplication commands

Table 15 — Approved services for Duplication commands

Service	Description	Approved Security Function
TPM2_Duplicate[9]	Duplicates a loaded object to a new parent object	RSA key transport ECC key generation ECC key agreement Symmetric key derivation (KBKDF) Symmetric key (un)wrapping Deterministic random bit generation Message digest
TPM2_Rewrap	Rewraps a duplicated object with a new parent key	RSA key transport ECC key generation ECC key agreement Symmetric key derivation (KBKDF) Symmetric key (un)wrapping Deterministic random bit generation
TPM2_Import	Imports a duplicated object in order that it can be loaded inside the TPM	RSA key transport ECC key agreement Symmetric key derivation (KBKDF) Symmetric key (un)wrapping Deterministic random bit generation Message digest HMAC signature generation

[9] Section 8 describes additional conditions for TPM2_Duplicate to meet FIPS 140-3 requirements for SSP entry and output methods

Table 16 — Non-Approved services for Duplication commands

Service	Description	Algorithm Accessed
TPM2_Duplicate	Duplicates a loaded object to a new parent object	RSA Key Transport with RSA key length equal to 1024 bits ECC Key generation with BN P-256 elliptic curve ECC key agreement with BN P-256 elliptic curve
TPM2_Rewrap	Rewraps a duplicated object with a new parent key	RSA Key Transport with RSA key length equal to 1024 bits ECC Key generation with BN P-256 elliptic curve ECC key agreement with BN P-256 elliptic curve ECC key agreement with Derived Asymmetric ECC Keys
TPM2_Import	Imports a duplicated object in order that it can be loaded inside the TPM	RSA Key Transport with RSA key length equal to 1024 bits ECC key agreement with BN P-256 elliptic curve ECC key agreement with Derived Asymmetric ECC Keys SSP input check: RSA key loading with RSA key length equal to 1024 bits ECC Key loading with BN P-256 elliptic curve HMAC key loading with Key Size < 112 bits

7.5.6 Chapter 14: Asymmetric Primitives

Table 17 — Approved services for Asymmetric primitives commands

Service	Description	Approved Security Function
TPM2_RSA_Encrypt	RSA Encryption	RSA key transport
TPM2_RSA_Decrypt	RSA Decryption	RSA key transport
TPM2_ECDH_KeyGen	Ephemeral key pair generation and Shared Secret Calculation	ECC key generation ECC Shared Secret Calculation
TPM2_ECDH_ZGen	Shared Secret Calculation	ECC Shared Secret Calculation

Table 18 — Non-Approved services for Asymmetric primitives commands

Service	Description	Non-Approved Security Function
TPM2_RSA_Encrypt	RSA Encryption	RSA Key Transport with RSA key length equal to 1024 bits RSA Key Transport with Non Approved Padding scheme NULL RSA Key Transport with Non Approved Padding scheme RSAES-PKCS-v1.5
TPM2_RSA_Decrypt	RSA Decryption	RSA Key Transport with RSA key length equal to 1024 bits RSA Key Transport with Non Approved Padding scheme NULL RSA Key Transport with Non Approved Padding scheme RSAES-PKCS-v1.5
TPM2_ECDH_KeyGen	Ephemeral key pair generation and Shared Secret Calculation with TPM ephemeral key and loaded public key	ECC Key generation with BN P-256 elliptic curve ECC Shared Secret Calculation with BN-P-256 elliptic curve
TPM2_ECDH_ZGen	Shared Secret Calculation with TPM static key and provided public key	ECC Shared Secret Calculation with BN-P-256 elliptic curve ECC Shared Secret Calculation with Derived Asymmetric ECC Key
TPM2_ZGen_2Phase	Ephemeral key pair derivation and Shared Secret Calculation with TPM ephemeral and static key and provided ephemeral and static key	Asymmetric ECC key derivation ECC Shared Secret Calculation with BN-P-256 elliptic curve ECC Shared Secret Calculation with Derived Asymmetric ECC Key (derived from counter or Derived Object)

7.5.7 Chapter 15: Symmetric primitives

Table 19 — Approved services for Symmetric primitives commands

Service	Description	Approved Security Function
TPM2_EncryptDecrypt	Symmetric encryption or decryption of user data	AES encryption/decryption
TPM2_EncryptDecrypt2	Symmetric encryption or decryption of user data	AES encryption/decryption
TPM2_Hash	Performs a hash operation on user data	Message digest HMAC signature generation
TPM2_HMAC	Performs a HMAC operation on user data	HMAC signature generation
TPM2_MAC	Performs a MAC operation on user data	HMAC signature generation

Table 20 — Non-approved services for Symmetric primitives commands

Service	Description	Algorithm Accessed
TPM2_HMAC	Performs a HMAC operation on user data	HMAC signature generation and verification with Key Size < 112 bits
TPM2_MAC	Performs a MAC operation on user data	HMAC signature generation and verification with Key Size < 112 bits

7.5.8 Chapter 16: Random Number Generator

Table 21 — Approved services for Random Number Generator commands

Service	Description	Approved Security Function
TPM2_GetRandom	Random number generation	Deterministic random bit generation
TPM2_StirRandom	Reseed random number	Deterministic random bit generation DRBG seeding

7.5.9 Chapter 17: Hash/HMAC/Event Sequences

Table 22 — Approved services for Hash/HMAC/Event Sequences commands

Service	Description	Approved Security Function
TPM2_HMAC_Start	HMAC session start	HMAC signature generation
TPM2_HashSequenceStart	Hash session start	Message digest
TPM2_SequenceUpdate	Sequence update	HMAC signature generation Message digest
TPM2_SequenceComplete	Sequence complete	HMAC signature generation Message digest
TPM2_EventSequenceComplete	Event sequence complete	HMAC signature generation Message digest

Table 23 — Non-Approved services for Hash/HMAC/Event Sequences commands

Service	Description	Non-Approved Security Function
TPM2_HMAC_Start	HMAC session start	HMAC signature generation and verification with Key Size < 112 bits
TPM2_SequenceUpdate	Sequence update	HMAC signature generation and verification with Key Size < 112 bits
TPM2_SequenceComplete	Sequence complete	HMAC signature generation and verification with Key Size < 112 bits
TPM2_EventSequenceComplete	Event sequence complete	HMAC signature generation and verification with Key Size < 112 bits

7.5.10 Chapter 18: Attestation commands

Table 24 — Approved services for Attestation commands

Service	Description	Approved Security Function
TPM2_Certify	Proves that an object with a specific Name is loaded in the TPM	ECC signature generation RSA signature generation HMAC signature generation Message digest Symmetric key derivation (KBKDF)
TPM2_CertifyCreation	Proves the association between an object and its creation data	
TPM2_Quote	Quotes PCR values	
TPM2_GetSessionAuditDigest	Returns a digital signature of the audit session digest	
TPM2_GetCommandAuditDigest	Returns the current value of the command audit digest	
TPM2_GetTime	Returns the current values of Time and Clock	
TPM2_CertifyX509	X.509 certificate generation	ECC signature generation RSA signature generation Message digest

Table 25 — Non-Approved services for Attestation commands

Service	Description	Algorithm Accessed
TPM2_Certify	Proves that an object with a specific Name is loaded in the TPM	RSA signature generation with RSA key length equal to 1024 bits RSA signature generation with SHA-1 ECDAA signature generation ECSCHNORR signature generation ECC Signature generation with Derived Asymmetric ECC Keys ECC signature generation with BN P-256 elliptic curve ECC signature generation with SHA-1 HMAC signature generation and verification with Key Size < 112 bits
TPM2_CertifyCreation	Proves the association between an object and its creation data	
TPM2_Quote	Quotes PCR values	
TPM2_GetSessionAuditDigest	Returns a digital signature of the audit session digest	
TPM2_GetCommandAuditDigest	Returns the current value of the command audit digest	
TPM2_GetTime	Returns the current values of Time and Clock	
TPM2_CertifyX509	X.509 certificate generation	RSA signature generation with RSA key length equal to 1024 bits RSA signature generation with RSA key length equal to 1024 bits ECDAA signature generation ECSCHNORR signature generation ECC Signature generation with Derived Asymmetric ECC Keys ECC signature generation with BN P-256 elliptic curve ECC signature generation with SHA-1

7.5.11 Chapter 19: Anonymous Attestation

Table 26 — Approved services for Anonymous Attestation commands

Service	Description	Non-Approved Security Function
TPM2_Commit	Ephemeral key pair derivation and Shared Secret Calculation with a TPM ephemeral key and static key and provided public keys	Asymmetric ECC key derivation ECC Shared Secret Calculation with BN-P-256 elliptic curve ECC Shared Secret Calculation with Derived Asymmetric ECC Key
TPM2_EC_Ephemeral	Ephemeral key pair derivation	Asymmetric ECC key derivation

7.5.12 Chapter 20: Signing and Signature Verification

Table 27 — Approved services for Signing and Signature Verification commands

Service	Description	Approved Security Function
TPM2_VerifySignature	Signature verification	RSA signature verification ECC signature verification HMAC signature verification HMAC signature generation
TPM2_Sign	Signature generation	RSA signature generation ECC signature generation HMAC signature generation

Table 28 — Non-Approved services for Signing and Signature Verification commands

Service	Description	Non-Approved Security Function
TPM2_VerifySignature	Signature verification	ECC Signature verification with Derived Asymmetric ECC Keys HMAC signature generation and verification with Key Size < 112 bits ECSCHNORR signature verification ECC signature verification with BN P-256 elliptic curve
TPM2_Sign	Signature generation	RSA signature generation with RSA key length equal to 1024 bits RSA signature generation with SHA-1 ECDAA signature generation ECSCHNORR signature generation ECC Signature generation with Derived Asymmetric ECC Keys ECC signature generation with BN P-256 elliptic curve ECC signature generation with SHA-1

7.5.13 Chapter 22: Integrity collection

Table 29 — Approved services for Integrity collection commands

Service	Description	Approved Security Function
TPM2_PCR_Extend	Updates the indicated PCR	Message digest
TPM2_PCR_Event	Updates the indicated PCR and reports a list of digests	Message digest

7.5.14 Chapter 23: Enhanced Authorization (EA) Commands

Table 30 — Approved services for Enhanced Authorization commands

Service	Description	Approved Security Function
TPM2_PolicySigned	Policy based on a signing key	Message digest RSA signature verification

		ECC signature verification HMAC signature verification HMAC signature generation
TPM2_PolicySecret	Policy based on an entity's authValue	Message digest HMAC signature generation
TPM2_PolicyTicket	Policy based on a ticket (produced by PolicySigned or PolicySecret)	Message digest HMAC signature generation
TPM2_PolicyOR	Policy enabling multiple authentication options	Message digest
TPM2_PolicyPCR	Policy based on PCR	Message digest
TPM2_PolicyLocality	Policy based on Locality	Message digest
TPM2_PolicyNV	Policy based on the contents of an NV Index	Message digest
TPM2_PolicyCounterTimer	Policy based on time	Message digest
TPM2_PolicyCommandCode	Policy based on command code	Message digest
TPM2_PolicyPhysicalPresence	Policy based on Physical Presence	Message digest
TPM2_PolicyCpHash	Policy bound to specific command with specific parameters and specific objects	Message digest
TPM2_PolicyNameHash	Policy bound to specific objects	Message digest
TPM2_PolicyDuplicationSelect	Policy limiting duplication to only a selected parent	Message digest
TPM2_PolicyAuthorize	Policy enabling a change in policy	Message digest HMAC signature verification
TPM2_PolicyAuthValue	Policy bound to the authValue of an authorized entity (requiring an HMAC session)	Message digest
TPM2_PolicyPassword	Policy bound to the authValue of an authorized entity (requiring a password session)	Message digest
TPM2_PolicyNvWritten	Policy based on the WRITTEN attribute of an NV Index	Message digest
TPM2_PolicyTemplate	Policy bound to a specific creation template	Message digest
TPM2_PolicyAuthorizeNV	Policy bound to a policy stored in an NV Index	Message digest

Table 31 — Non-Approved services for Enhanced Authorization commands

Service	Description	Non-Approved Security Function
TPM2_PolicySigned	Policy based on a signing key	ECC Signature verification with Derived Asymmetric ECC Keys HMAC signature generation and verification with Key Size < 112 bits ECSCHNORR signature verification ECC signature verification with BN P-256 elliptic curve
TPM2_PolicySecret	Policy based on an entity's authValue	HMAC signature generation and verification with Key Size < 112 bits
TPM2_PolicyTicket	Policy based on a ticket (produced by PolicySigned or PolicySecret)	HMAC signature generation and verification with Key Size < 112 bits
TPM2_PolicyAuthorize	Policy enabling a change in policy	HMAC signature generation and verification with Key Size < 112 bits

7.5.15 Chapter 24: Hierarchy commands

Table 32 — Approved services for Hierarchy commands

Service	Description	Approved Security Function
TPM2_CreatePrimary	Creates a Primary Object	AES key generation ECC key generation RSA key generation HMAC key generation DRBG seeding Deterministic random bit generation Message digest HMAC signature generation
TPM2_ChangePPS	Changes the current platform primary seed (PPS)	Deterministic random bit generation
TPM2_ChangeEPS	Changes the current endorsement primary seed (EPS)	Deterministic random bit generation
TPM2_Clear	Removes all TPM context associated with a specific Owner	Deterministic random bit generation

Table 33 — Non-Approved services for Hierarchy commands

Service	Description	Algorithm Accessed
TPM2_CreatePrimary	Creates a Primary Object	ECC Key generation with BN P-256 elliptic curve HMAC key generation with Key Size < 112 bits RSA key generation with RSA key length equal to 1024 bits

7.5.16 Chapter 28: Context Management

Table 34 — Approved services for Context Management commands

Service	Description	Approved Security Function
TPM2_ContextSave	Save the context of an object, object sequence or session	Symmetric key derivation (KBKDF) Symmetric key (un)wrapping
TPM2_ContextLoad	Reload a context	Symmetric key derivation (KBKDF) Symmetric key (un)wrapping SSP input check: RSA key loading with RSA key length equal to 1024 bits ECC Key loading with BN P-256 elliptic curve HMAC key loading with Key Size < 112 bits

7.5.17 Chapter 29: Clocks and timers

No TPM clock or timer commands provide approved services

7.5.18 Chapter 30: Capability commands

No TPM capability commands provide approved services

7.5.19 Chapter 31: Non-volatile Storage

Table 35 — Approved services for Non-volatile Storage commands

Service	Description	Approved Security Function
TPM2_NV_ReadPublic	Read the public area and name of an NV Index	Message digest
TPM2_NV_Extend	Extend data to an NV Index	Message digest
TPM2_NV_Certify	Certify the contents of an NV Index	Message digest RSA signature generation ECC signature generation HMAC signature generation

Table 36 — Non-Approved services for Non-volatile Storage commands

Service	Description	Non-Approved Security Function
TPM2_NV_Certify	Certify the contents of an NV Index	ECC signature generation with SHA-1 RSA signature generation with SHA-1 RSA signature generation with RSA key length equal to 1024 bits ECC Signature generation with Derived Asymmetric ECC Keys ECDAA signature generation ECSCHNORR signature generation ECC signature generation with BN P-256 elliptic curve

7.5.20 Chapter 32: Attached components commands

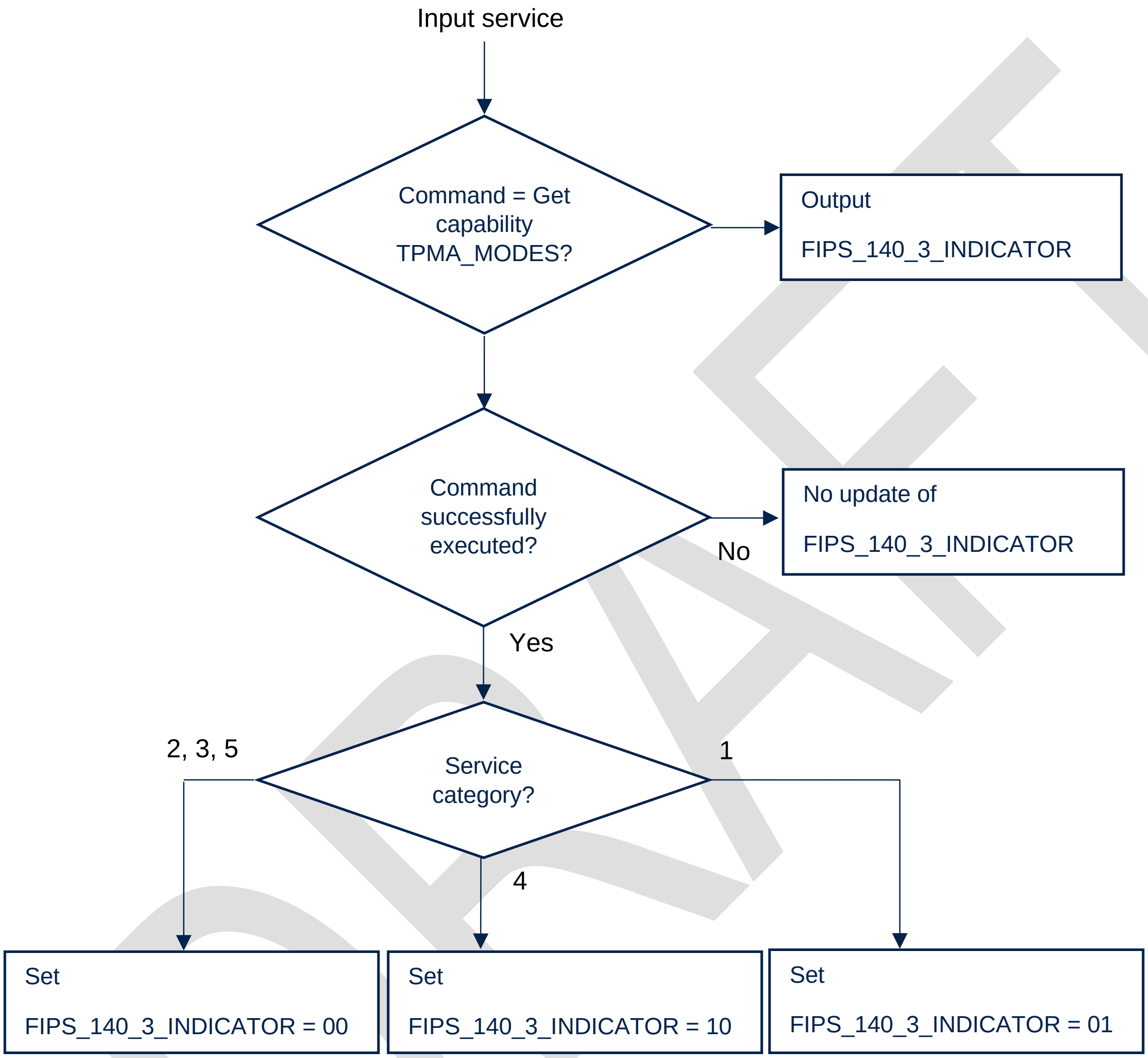
No TPM Attached Components commands provide approved services.

No TPM Attached Components commands provide non-approved services.

7.6 FIPS indicator flowchart

Figure 1 illustrates the flow to set the FIPS indicator.

Figure 1 — FIPS indicator flowchart


Input service
Command = Get capability TPMA_MODES?
Output FIPS_140_3_INDICATOR
Command successfully executed?
No
No update of FIPS_140_3_INDICATOR
Yes
Service category?
2, 3, 5
1
4
Set FIPS_140_3_INDICATOR = 00
Set FIPS_140_3_INDICATOR = 10
Set FIPS_140_3_INDICATOR = 01

7.7 Command sequence example

Table 37 below provides an example of the execution of a sequence of TPM commands, the impact on the FIPS indicator reported by the command TPM2_GetCapability, and the value of the indicator stored in the TPM state after command execution as described in Figure 1.

Table 37 — Example of TPM commands execution and FIPS indicator value

Event	**Service category**	**Reported FIPS_140_3_INDICATOR**	**Internal FIPS_140_3_INDICATOR value after command execution**
power up			00b (default value)
Service	1		01b
GetCapability (TPMA_Modes)	2	01b	00b
Service	5		00b
GetCapability (TPMA_Modes)	2	00b	00b
Service	4		10b
GetCapability (TPMA_Modes)	2	10b	00b
GetCapability (TPMA_Modes)	2	00b	00b
Service	2		00b
Service	1		01b
Service (not executed successfully)	3		01b
GetCapability (TPMA_Modes)	2	01b	00b
Service	4		10b
Service	1		01b
Service	2		00b
GetCapability (TPMA_Modes)	2	00b	00b
GetCapability (TPMA_Modes)	2	00b	00b

8 SSP Validation requirements

8.1 Public Key Validation

SP800-56Br2 section 6.4.2.1 requires the recipient of an RSA key to obtain assurance of public-key validity using one or more of the following methods:

1. Partial Public-Key Validation
2. TTP (trusted third party) Partial Public-Key Validation
3. TTP (trusted third party) Key-Pair Validation

SP800-56Br2 section 6.4.2.2 specifies the Partial Public-Key Validation for RSA, which refers to SP800-89 section 5.3.3 for the specific requirements, which are listed below:

1. The length of the modulus is one of the specified values in FIPS 186-5
2. The value of the public exponent is in the valid range, as specified in FIPS 186-5
3. The modulus and the public exponent are odd numbers
4. The modulus is composite, but not a power of a prime
5. The modulus has no factors smaller than 752. Testing for additional factors is allowed

If the TPM performs Partial Public-Key Validation, the TPM performs these checks for every RSA key, which is identified as not having been generated by the TPM itself, during the import or before the first use of the key.

The service indicator does not reflect whether Partial Public-Key Validation has been performed by the TPM.

9 SSP entry and output Methods

This section explains how FIPS 140-3 requirements for SSP entry and output methods are implemented in the TPM Library specification.

Based on FIPS 140-3 IG 9.5.A, a CSP Entry format for a TPM is categorized according to Table 2 as

- Distribution: Manual
- Entry (Input/Output): Electronic

Up to level 2, the IG allows that SSPs established using manual distribution and electronically input or output to a cryptographic module may be input or output in plaintext.

Per ISO/IEC 19790:2012, section 7.9.5, "*To prevent the inadvertent output of sensitive information, two independent internal actions **shall [9.16]** be required in order to output any plaintext CSP*."[10]

Therefore, the two following independent internal actions are necessary when authorizing a TPM2_Duplicate command to output the private part of a key in plaintext form

- Verification of the encryptedDuplication attribute of the key to be duplicated: encryptedDuplication attribute needs to be set to 0 and
- Verification of the handle of the new parent of the key to be duplicated: the new handle needs to be set to the NULL handle.

Both conditions need be met to authorize outputting the private part of the key in plaintext form.

[10] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.

10 Self-tests

Section 10.1 lists the summary of the FIPS 140-3 requirements applicable to self-tests from ISO/IEC 19790, ISO/IEC 24759, SP800-140C rev2 and clarifications provided in FIPS 140-3 Implementation Guidance.
In that section 10.1, the term “module” used in FIPS 140-3 terminology should be interpreted to mean a TPM implementation.
Section 10.2 provides ways in which TPM 2.0 implementations can address FIPS 140-3 requirements.

10.1 FIPS 140-3 requirements

10.1.1 ISO/IEC 19790 and ISO/IEC 24759

10.1.1.1 Applicable to all security levels (1, 2, 3, 4)

As stated in ISO/IEC19790:2012 section 7.10.1, [11]

All self-tests shall [10.01] be performed, and determination of pass or fail shall [10.02] be made by the module, without external controls, externally provided input text vectors, expected output results, or operator intervention or whether the module will operate in an approved or non-approved mode.

The pre-operational self-tests shall [10.03] be performed and passed successfully prior to the module providing any data output via the data output interface.

Conditional self-tests shall [10.04] be performed when an applicable security function or process is invoked (i.e. security functions for which self-tests are required).

If a self-test fails, the module shall enter an error state and output an error indicator. No cryptographic operation, data or control output shall be performed while the module is in error state.

Self-tests are split into two categories:

- The pre-operational self-tests that shall be passed successfully prior to any data output by the module
- The conditional self-tests that shall be performed when an applicable cryptographic algorithm or process is invoked

Per ISO/IEC 19790:2012, section 7.10.2.1[12]:

A cryptographic module shall [10.16] perform the following pre-operational self-tests, as applicable:

- *pre-operational software/firmware integrity test;*
- *pre-operational bypass test; and*
- *pre-operational critical functions test.*

Per ISO/IEC 19790:2012, section 7.10.2.2[13]:

All software and firmware components within the cryptographic boundary shall [10.17] be verified using an approved integrity technique or EDC satisfying the requirements defined in 7.5.

[11] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.
[12] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.
[13] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.

Per ISO/IEC 19790:2012, section 7.10.3.1[14]:

Conditional self-tests shall [10.25] be performed by a cryptographic module when the conditions specified for the following tests occur: Cryptographic Algorithm Self-Test, Pair-Wise Consistency Test, Software/Firmware Load Test, Manual Entry Test, Conditional Bypass Test and Conditional Critical Functions Test.

The conditional tests are:

- Cryptographic algorithm tests:
  A test (KAT, comparison test or fault detection test) shall be conducted for all approved cryptographic algorithms. The list of algorithms referenced in Annex C of ISO/IEC 19790 is superseded by the list in NIST SP800-140C. At a minimum, the smallest approved key length, modulus size or curve shall be tested. If multiple modes are specified (for example CFB, CBC), only one mode can be self-tested. Forward and inverse functions of an algorithm shall be tested.
- Cryptographic pair-wise consistency tests:
  A PCT shall be performed for every generated asymmetric key pair.
- Software/firmware load test:
  This self-test shall be performed prior to loading external software or firmware on the security module. It consists in using an approved authentication technique to verify the validity of the software/firmware loaded. The authentication key shall be loaded on the module before software/firmware loading.
- Manual entry test (applies only to SSPs manual entry)
- Bypass test (if the module implements a bypass capability, which is not the case for TPM)
- Critical functions test (other security functions critical to the secure operation of the cryptographic module and defined in documentation)

The module shall also permit operators to initiate the pre-operational or conditional self-tests on demand for periodic testing of the module.

10.1.2 FIPS 140-3

10.1.2.1 NIST SP800-140C

SP800-140C rev2 identifies CMVP-approved security functions. It supersedes security functions identified in ISO/IEC 19790 and ISO/IEC 24759.

10.1.2.2 Implementation Guidance

IG 10.3.A details the type and specificities of the self-tests that are required for each approved algorithm. The requirements applicable to TPM2.0 are detailed in section 10.2 of this document.

IG 10.3.B describes the conditions when there is no requirement to perform a self-test on an embedded core cryptographic implementation.

IG 10.3.C details the requirements relative to the SSPs manual entry.

10.2 Implementation of Self-Tests

10.2.1 Pre-operational self-tests

The list of pre-operational self-tests is given in Table 38 below.

[14] ©ISO. This material is reproduced from ISO/IEC 19790:2012 with permission of the American National Standards Institute (ANSI) on behalf of the International Organization for Standardization. All rights reserved.

Table 38 — Pre-operational self-tests

Self-test	Algorithm	Notes
Software/firmware integrity tests	EDC (16-bit minimum), MAC or digitial signature	The three algorithms are usable for hardware modules at all security levels. For software or firmware modules and hybrid modules, if firmware is located outside of the security boundary, only MAC or digital signatures can be used from security level 2 to level 4.
Bypass test	-	Implementation dependent and out of scope of this guidance
Critical functions test	-	Implementation dependent and out of scope of this guidance

10.2.2 Conditional self-tests

10.2.2.1.1 Cryptographic algorithm tests

Table 39 below identifies the cryptographic algorithms that have to be tested with conditional tests in a PC Client specific TPM2.0 module compliant with PTP 1.04 or PTP 1.05. Table 39 may be augmented by a TPM vendor to reflect the scope of algorithms supported by their TPM 2.0 product implementation. It is also up to vendors to detail the size of the input data, keys or seeds used in their implementation of the self-tests.

Table 39 — Conditional self-tests

Algorithm	Self-test details	Reference standard	Notes
SHA1	KAT	FIPS 180-4	SHA1 of known data. Comparison of output with expected 160-bit digest value.
SHA256	KAT	FIPS 180-4	SHA256 of known data. Comparison of output with expected 256-bit digest value.
SHA384	KAT	FIPS 180-4	SHA384 of known data. Comparison of output with expected 384-bit digest value.
HMAC	KAT	FIPS 198-1	HMAC (using either SHA1, SHA256 or SHA384) of known data with a known key. Comparison of output with expected MAC value.
KDA	KAT	NIST SP800-56A	KDF (using either SHA1, SHA256 or SHA384) of known secret. This KAT can be grouped with ECDH KAT (see notes for the ECDH algorithm).
KDF	KAT	NIST SP800-108	KDF (using either HMAC-SHA1, HMAC-SHA256 or HMAC-SHA384) of known secret. Comparison of output with expected derived value.
DRBG	KAT	NIST SP800-90A	Instantiate, reseed and generate APIs must be tested. The recommendation of this guidance is to group APIs into a single KAT that comprises instantiating and reseeding the DRBG with known seeds and comparing the output of the generateAPI with an expected value.
AES	KAT	FIPS 197, NIST SP800-38A	AES encryption of known plaintext with a known key (128-bit, 192-bit or 256-bit) and IV. Comparison of the output with an expected ciphertext value. AES decryption of produced ciphertext with the same key and IV as for encryption. Comparison of the output with the known plaintext.
ENT	Health-tests	NIST SP800-90B	RCT and APT tests from SP800-90B (sections 4.4.1 and 4.4.2). Alternatively, developer-defined tests might be used instead of (or in addition to) the approved tests, provided the choice is justified.
RSA	KAT	FIPS 186-4	RSA signature generation with known data and key (2048-bit or 3072-bit) and either RSASSA-PKCS-v1_5 scheme or RSA-PSS, the latter with a fixed randomization parameter. Comparison of the output with a known signature. RSA signature verification of a generated signature.
ECDSA	KAT	FIPS 186-4	ECDSA signature generation with known data, key (256-bit or 384-bit) and a fixed randomization parameter. Comparison of the output with a known signature. ECDSA signature verification of the generated signature with the same data and key.

ECDH	KAT	NIST SP800-56Ar3	A shared secret is generated with ECDH (primitive “Z” computation) with a known point P and known private key on NIST P-256 or P-384 curve. Comparison of the output with a known shared secret. Additionaly, KDA algorithm testing can be grouped with this KAT to produce a known key from the shared secret and compare it to a known value.

10.2.2.2 Cryptographic pair-wise consistency tests

Table 40 identifies the cryptographic pair-wise consistency tests that are mandatory according to the FIPS 140-3 specification.

Table 40 — Pair-wise consistency tests

Algorithm	Self-test details	Reference standard	Notes
RSA key generation (2048-bit or 3072-bit length)	PCT	FIPS 186-4	If the generated key is for encryption (the sign attribute of the key is not set), RSA encryption using RSAES-OAEP scheme is performed with the generated key and known plaintext. The encryption output is compared to the plaintext to check that they are different. Then decryption is performed and the decryption output is compared to the known plaintext to check that they are identical. If the generated key is for signing (the sign attribute is set), an RSA signature is generated using RSASSA-PKCS-v1_5 or RSASSA-PSS with a fixed randomization parameter and fixed data, then verified to determine that PCT is successful.
ECC key generation (NIST P-256 or NIST P-384 curve)	PCT	FIPS 180-4 NIST SP800-56Ar3	If the generated key is for secret sharing (the sign attribute of the key is not set), the ECC public key is computed again from the generated private key and compared to the initially generated ECC public key (see section 5.6.2.1.4 of SP800-56Ar3). If the generated key is for signing (the sign attribute is set), an ECC signature is generated using ECDSA scheme with a fixed randomization parameter and fixed data, then verified to determine that PCT is successful.

10.2.2.3 Software/firmware load test

Implementation of this self-test is vendor dependent. This guidance recommends satisfying the requirements listed in section 10.1.1.1

10.2.2.4 Manual entry test

Implementation of this self-test is vendor dependent.

10.2.2.5 Bypass test

Implementation of this self-test is vendor dependent.

10.2.2.6 Critical functions test

Implementation of this self-test is vendor dependent.

10.2.3 Periodic self-tests

The periodic self-tests requirement is covered by using TPM2_SelfTest (fullTest = YES) command on-demand.