STMICROELECTRONICS

Trusted Platform Module ST33TPHF2ESPI

ST33HTPH2E28AAF0 / ST33HTPH2E32AAF0 /
ST33HTPH2E28AAF1 / ST33HTPH2E32AAF1

FIPS 140-2 Security Policy
Level 1

Firmware revision: 49.00
HW version: ST33HTPH revision A

Date: 2017/05/17
Document Version: 01-07

NON-PROPRIETARY DOCUMENT

Table of Contents

1 MODULE DESCRIPTION

1.1 Definition

The ST33TPHF2ESPI Trusted Platform Module is a fully integrated security module designed to be integrated into personal computers and other embedded systems. The security module is used primarily for cryptographic key generation, key storage and key management as well as generation and secure storage for digital certificates.

The TPM is a single chip cryptographic HW module as defined in **[FIPS 140-2]**. The single silicon chip is encapsulated in a hard, opaque, production grade integrated circuit (IC) package.

The cryptographic boundary is defined as the perimeter of the IC package. The security module supports SPI interface compliant with the Trusted Computing Group (TCG) specification for PC Client [PTP 0.43]. The HW and FW cryptographic boundaries are indicated in §1.4 of the current document.

The security module implements both version 1.2 and 2.0 of the Trusted Computing Group (TCG) specification for Trusted Platform Modules (TPM).

1.2 Module identification

The hardware and firmware versions covered by the FIPS evaluation are identified as follow:

- Hardware version: ST33HTPH revision A
- Firmware version: 49.00 (SPI)

FW version can be retrieved via response to the command TPM_GetCapability (TPM1.2) and TPM2_GetCapability (TPM2.0) with property set to TPM_PT_FIRMWARE_VERSION_1.

The cryptographic services are provided by the cryptographic library “NesLib 5.1 for ST33”.

The product is manufactured in two packages:

- TSSOP28
  - TSSOP 28-pin
  - 4.4 x 9.7 mm
- VQFN32
  - Very thin pitch Quad pack no-lead 32-pin
  - 5 x 5 mm

The security module is available in the following configurations:

Table 1: Security module configurations

	Module configuration			
Product name / HW version	ST33TPHF2ESPI/ ST33HTPH revision A			
Package	TSSOP28	VQFN32	TSSOP28	VQFN32
Part number	ST33HTPH2E28 AAF0	ST33HTPH2E32 AAF0	ST33HTPH2E28 AAF1	ST33HTPH2E32 AAF1
Marking	P68HAAF0		P68HAAF1	
FW version	49.00			

P68HAAF0 corresponds to the module configured by default in TPM1.2 execution mode. P68HAAF1 corresponds to the module configured by default in TPM2.0 execution mode.

Figure 1: Picture of the Cryptographic Module (TSSOP28) – Marking

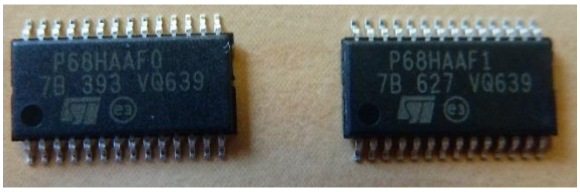


Figure 2: Picture of the Cryptographic Module (VQFN32) – Marking

1.3 Pinout description

The pin layouts for the ST33TPHF2ESPI are shown in Figure 3 and in Figure 4: VQFN32 Pinout Diagram.

Figure 3: TSSOP28 Pinout Diagram

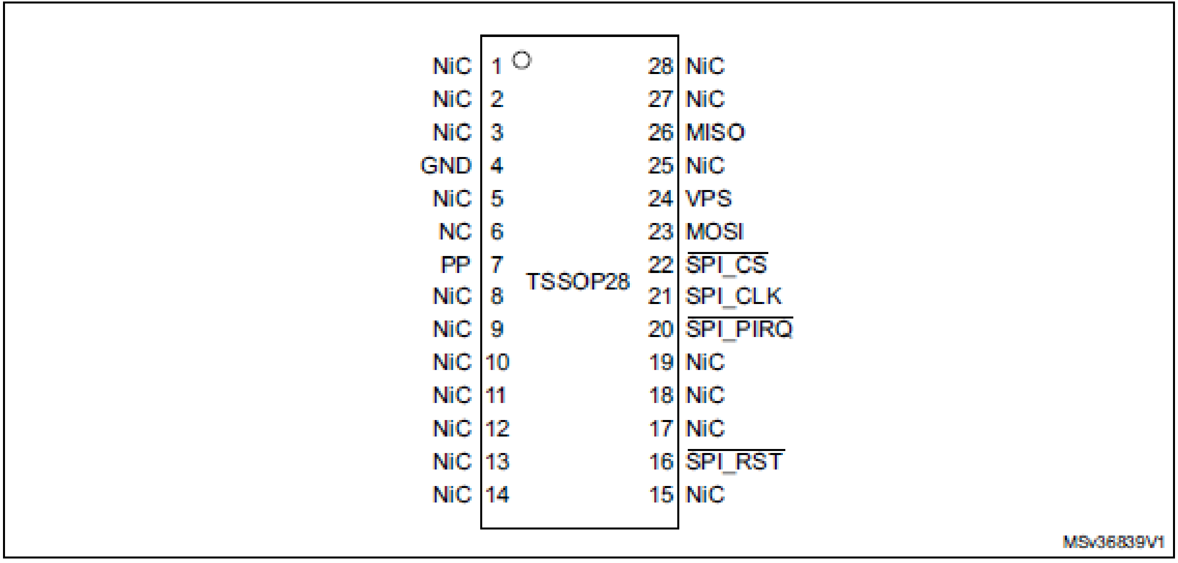
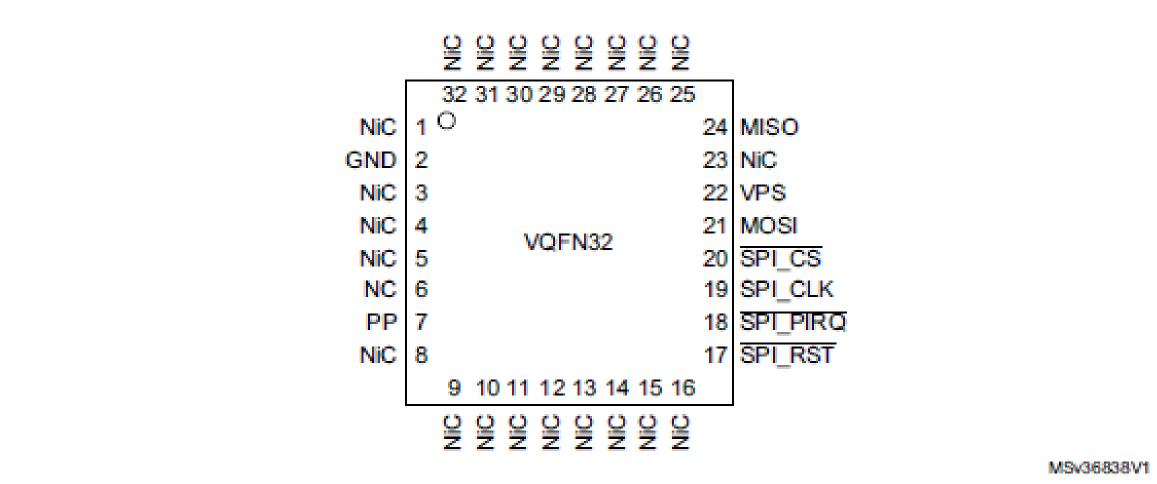


Figure 4: VQFN32 Pinout Diagram



Next table gives a description of the products pins.

Table 2: ST33TPHF20E Pin definition (SPI configuration)

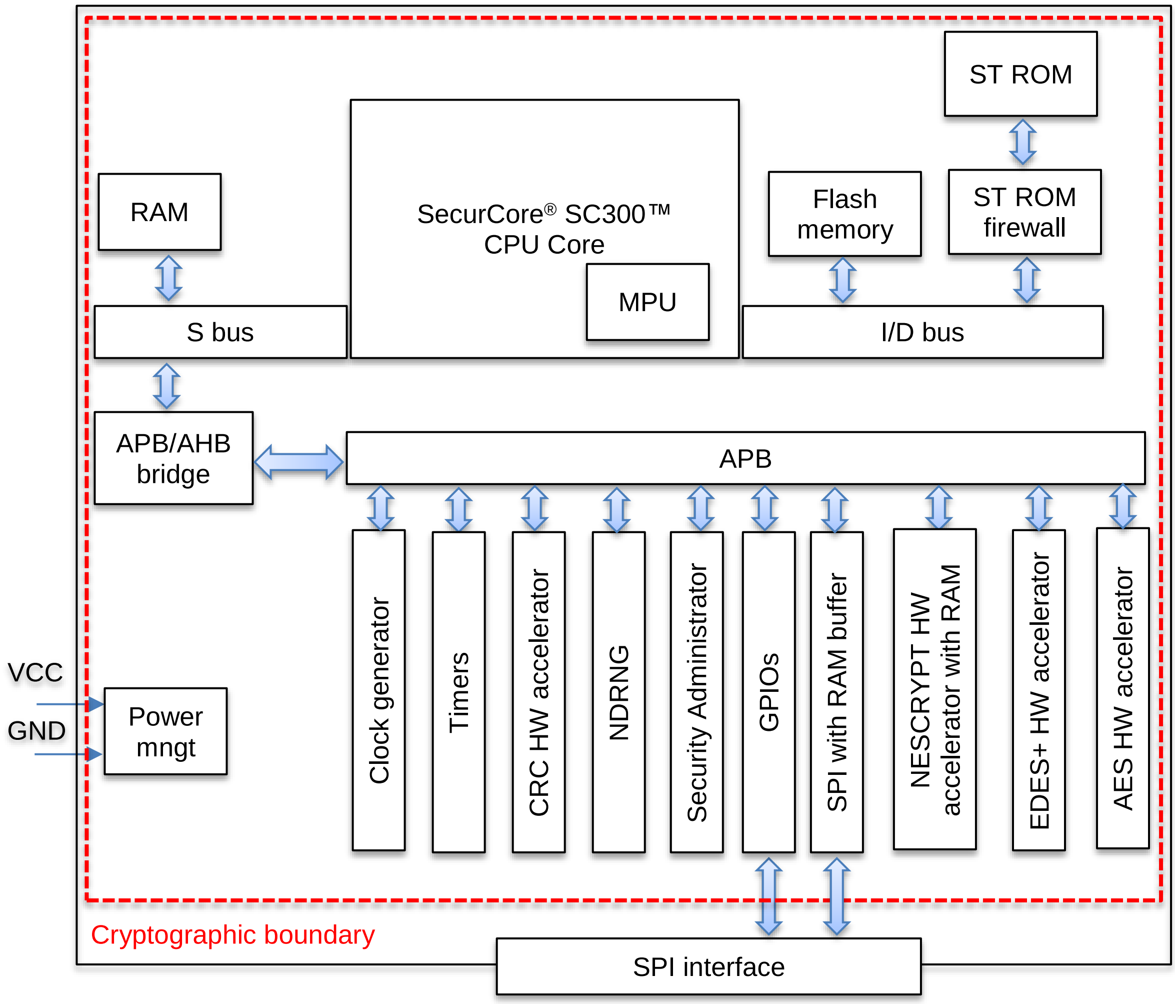
Signal	Type	Description
VPS	Input	**Power supply**. This pin must be connected to 1.8V or 3.3V DC power rail supplied by the motherboard.
GND	Input	GND has to be connected to the main motherboard ground.
$\overline{\text{SPI\_RST}}$	Input	**SPI Reset** used to re-initialize the device
MISO	Output	**SPI** Master Input, Slave Output (output from slave)
MOSI	Input	**SPI** Master Output, Slave Input (output from master)
SPI_CLK	Input	**SPI** serial clock (output from master)
$\overline{\text{SPI\_CS}}$	Input	**SPI** slave select (active low; output from master)
$\overline{\text{SPI\_PIRQ}}$	Output	**SPI IRQ** used by TPM to generate an interrupt
PP	Input	**Physical presence**, active high, internal pull-down. Used to indicate Physical Presence to the TPM.
NiC	-	**Not internally connected:** not connected to the die. May be left unconnected but no impact on TPM if connected.
NC	-	**Not Connected:** connected to the die but not usable. May be left unconnected. Internal pull-down.

1.4 Block diagrams

A block diagram of the hardware ST33HTPH (with its associated cryptographic boundary) is provided in Figure 5. TPM is composed of:

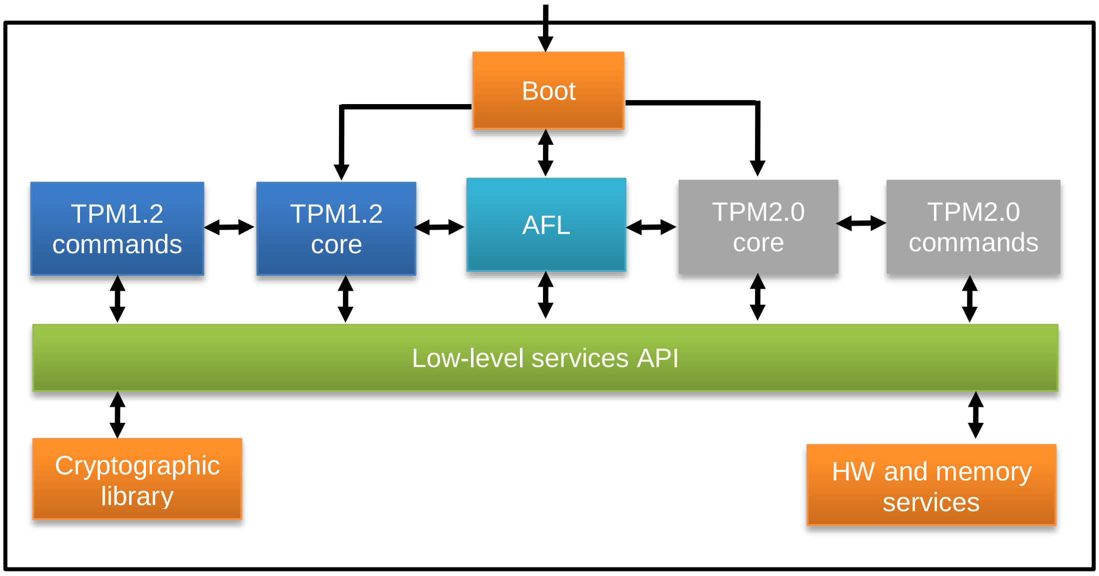
- A SecurCore® SC300™ CPU core including a MPU (Memory Protection Unit)
- Memories (RAMs, Flash and ROM)
- HW accelerators for CRC (16 and 32-bits) and cryptographic operations (symmetric with EDES+ and AES and asymmetric with NESCRYPT)
- A clock generator and three 16-bit timers
- NDRNG (non-deterministic random bit generator)
- SPI master/slave block
- A security administration block dedicated to chip security configuration and alarms detection
- FW and data stored in the memory areas

Figure 5: ST33HTPH block diagram



A block diagram of the TPM FW is provided in Figure 6: TPM FW block diagram.

Figure 6: TPM FW block diagram



TPM FW is composed of:

- Non-upgradable code blocks located in ROM & flash memories (depicted in orange)
  - Boot code
  - Cryptographic library
  - HW and memory low-level services
- Upgradable code blocks via secure field upgrade mechanism (blue and green boxes)
  - Application flash loader (AFL) in charge of TPM field upgrade
  - TPM1.2 core
  - TPM1.2 commands code
  - TPM2.0 core
  - TPM2.0 commands code
  - Low-level services API (incl. cryptographic services, memory management, …)

1.5 Security levels

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2.

Table 3: Module Security Level Specification

Security Requirements Section	**Level**
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	N/A
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1
Overall	**1**

1.6 Cryptographic functions

The security module supports the following cryptographic algorithms (both approved and non-approved). Algorithm certificate numbers for each approved algorithm are listed below. All algorithms, keys size or curve lengths listed below are part of services offered by the module.

Table 4: Approved algorithms

CAVP Cert #	Algorithm	Standard	Mode / Method	Key lengths, curves or moduli	Use
2342	RSA	FIPS 186-4	SHA-256, RSASSA-PKCS-v1.5	2048	Digital signature generation
		FIPS 186-4	SHA-1, SHA-256, RSASSA-PKCS-v1.5, RSASSA-PSS	2048	Digital signature verification
		FIPS 186-4	Appendix C3.1	2048	Key generation
1045	CVL RSADP	FIPS 186-4	RSA decryption primitive	2048	Key transport
1025	ECDSA	FIPS 186-4	SHA-256	P-224, P-256	Digital signature generation
		FIPS 186-4	SHA-1, SHA-256	P-224, P-256	Digital signature verification
		FIPS 186-4	Appendix B.4.2	P-224, P-256	Key generation
110	KAS EC-DH	SP 800-56A	ECC	P-224, P-256	Key agreement
2870	HMAC (single call)	FIPS 198-1	SHA-1, SHA-256	160, 256	Message authentication
2878	HMAC (sequence)	FIPS 198-1	SHA-1, SHA-256	160, 256	Message authentication
123	KBKDF	SP 800-108	CTR		Key derivation
1044	CVL TPM[1]	SP 800-135	HMAC SHA-1		Key derivation
1361	DRBG	SP 800-90A	HASH_based		Deterministic random bit generation
4338	AES	FIPS 197, SP 800-38A	ECB, CFB, OFB, CBC, CTR	128, 192, 256	Data encryption/decryption
2345	Triple-DES	SP 800-67, SP 800-38A	TECB, TCBC, TCFB64, TOFB, CTR	192	Data encryption/decryption
NA	KTS (AES cert #4338 + HMAC cert #2870)	SP 800-38F	CFB	128, 256	Key transport
3539	SHS	FIPS 180-4	SHA-1, SHA-256		Message digest
Vendor affirmation	CKG	SP800-133 (per IG D.12)	Direct generation, Generation		Key generation[2]

[1] TPM key establishment protocol that uses TPM KDF has not been reviewed or tested by the CAVP and CMVP (IG D.11)

[2] Symmetric keys and seeds used for generating the asymmetric keys are either generated by using KBKDF method (TPM2.0), DRBG (unmodified output for TPM2.0 & TPM1.2) or CVL TPM (TPM1.2). Methods are detailed per CSPs in tables Table 16: Keys and CSPs list and Table 20: Keys and CSPs list.

Table 5: Allowed algorithms

Algorithm	Caveat	Use
RSA	Key length = 1024 bits	Digital signature verification
RSA key wrapping	Key length = 2048 bits	Key establishment
SHA-1	NA	Digital signature verification
NDRNG	NA	Seed or reseed DRBG 800-90A (with approximatively 366 bits of entropy). Generate random numbers not dedicated to be used as cryptographic material.

Table 6: Non-approved algorithms

Algorithm	Use
RSA (key length = 1024 bits)	Key and digital signature generation
SHA-1	Digital signature generation
ECSchnorr	Digital signature generation and verification
ECDAA	Digital signature generation
MGF1	Encryption/decryption

1.7 Modes of Operation

At a given time, TPM is configured to process commands according to TPM1.2 or TPM2.0 standard. To operate in FIPS approved mode, TPM shall follow the recommendations listed in the paragraphs below according to the current mode (TPM1.2 or TPM2.0). Switch between TPM1.2 and TPM2.0 (and vice versa) is possible via execution of TPM_SetMode (TPM1.2) or TPM2_SetMode (TPM2.0) command followed by a reset of the TPM.

This security policy only applies to the security module when TPM operator follows the recommendations listed below to use the TPM in a FIPS approved mode of operation in TPM1.2 and TPM2.0 modes.

1.7.1 *TPM1.2 approved mode*

TPM1.2 supports 2 sequential approved modes of operation. To operate in full FIPS approved mode in TPM1.2:

- TPM_SelfTestFull command must be executed to switch from limited approved mode to full operational approved mode.
- FIPS mode recommendations listed in §1.7.1.3 must be followed.

1.7.1.1 *Limited approved mode*

This mode is the default mode when TPM starts. This mode is limited to a subset of TPM services.

Table 7: TPM1.2 limited approved mode

Properties	Description
Definition	Transient mode of operation when TPM is power-up and before TPM_SelfTestFull execution
Configuration	No configuration required
Services available	List of available services is indicated in last column of Table 17: Command support table
Algorithms used	SHA-1 & SHA-256
CSPs used	List of CSPs that might be accessed in this mode is indicated in Table 16: Keys and CSPs list
Self-tests	SHS / HW integrity / FW integrity / NDRNG

1.7.1.2 *Full operational approved mode*

This mode is the full FIPS approved mode of operation.

Table 8: TPM1.2 full operational approved mode

Properties	Description
Definition	Full approved mode of operation
Configuration	TPM_SelfTestFull execution
Services available	All services listed in Table 17: Command support table
Algorithms used	All supported algorithms (cf. §1.6)
CSPs used	All CSPs indicated in Table 16: Keys and CSPs list
Self-tests	SHS / HMAC / AES / DRBG / KDF / RSA / HW integrity / FW integrity / NDRNG

1.7.1.3 *FIPS mode recommendations*

To use TPM in a full approved FIPS 140-2 mode (valid for both modes), TPM user **shall**:

- Follow the procedure to switch from limited to full operational mode
- Use the following commands with the indicated restrictions (Indicated parameters refer to **[TPM1.2 Part3 r1.16]**. Indicated values refer to **[TPM1.2 Part2 r1.16]**.):
    - TPM_TakeOwnership: authDataUsage field of srkParams parameter must be different from TPM_AUTH_NEVER.
    - TPM_CreateWrapKey: size of key to be created must be strictly greater or equal to 2048 bits (keySize field of keyInfo parameter). authDataUsage field of keyInfo parameter must be different from TPM_AUTH_NEVER.
    - TPM_LoadKey / TPM_LoadKey2: size of key to be loaded must be strictly greater or equal to 2048 bits (keySize field of keyInfo parameter). authDataUsage field of keyInfo parameter must be different from TPM_AUTH_NEVER.
    - TPM_CMK_CreateKey: size of key to be created must be strictly greater or equal to 2048 bits (keySize field of keyInfo parameter). authDataUsage field of keyInfo parameter must be different from TPM_AUTH_NEVER.
    - TPM_MakeIdentity: authDataUsage field of idKeyParams parameter must be different from TPM_AUTH_NEVER.
    - TPM_EstablishTransport: If transAttributes field of transPublic parameter is equal to TPM_TRANSPORT_ENCRYPT, algId field must be different from TPM_ALG_MGF1.
- Use TPM_OSAP for authentication sessions with TPM_ET_AES128_CTR ADIP encryption scheme for commands listed in Table 18: Encrypted methods for secret and private keys input and marked as using AES_CTR to input or output CSPs.
- Use SHA-256 hash algorithm for digital signature generation. It concerns the following services:
    - TPM_Sign
- Not use services that don't meet FIPS 140-2 criteria:
    - TPM_DAA_Join (use of MGF1 as encryption scheme)
    - TPM_DAA_Sign (use of MGF1 as encryption scheme)
    - TPM_CertifyKey (signature generation using SHA1)
    - TPM_CertifyKey2 (signature generation using SHA1)
    - TPM_Quote (signature generation using SHA1)
    - TPM_Quote2 (signature generation using SHA1)
    - TPM_TickStampBlob (signature generation using SHA1)
    - TPM_ReleaseTransportSigned (signature generation using SHA1)

If operator does not strictly follow the FIPS approved mode recommendations, TPM is considered as being in a FIPS non-approved mode of operation.

To use the TPM in a FIPS approved mode if it was previously used in a FIPS non-approved mode, the operator shall:

- Zeroize all data listed in Table 16: Keys and CSPs list that could potentially be reused as CSPs in FIPS approved mode

To use the TPM in a FIPS non-approved mode if it was previously used in a FIPS approved mode, the operator shall:

- Zeroize all CSPs listed in Table 16: Keys and CSPs list that could potentially be used by FIPS non-approved algorithms in FIPS approved mode

1.7.2 *TPM2.0 approved mode*

TPM2.0 supports 2 sequential approved modes of operation. To operate in full FIPS approved mode in TPM2.0:

- TPM2_SelfTest (full=YES) command must be executed to switch from limited approved mode to full operational approved mode.
- FIPS mode recommendations listed in §1.7.2.3 must be followed.

1.7.2.1 *Limited approved mode*

This mode is the default mode when TPM starts. This mode is limited to a subset of TPM services.

Table 9: TPM2.0 limited approved mode

Properties	Description
Definition	Transient mode of operation when TPM is power-up and before TPM2_SelfTest(full=YES) execution
Configuration	No configuration required
Services available	List of available services is indicated in last column of Table 17: Command support table.
Algorithms used	SHA / HMAC / AES / DRBG / KDF / Triple-DES
CSPs used	List of CSPs that might be accessed in this mode is indicated in Table 20: Keys and CSPs list.
Self-tests	SHS / HMAC / AES / DRBG / KDF / Triple-DES / HW integrity / FW integrity / NDRNG

1.7.2.2 *Full operational approved mode*

This mode is the full FIPS approved mode of operation.

Table 10: TPM2.0 full operational approved mode

Properties	Description
Definition	Full approved mode of operation
Configuration	TPM2_SelfTest(full=YES) execution
Services available	All services
Algorithms used	All supported algorithms (cf. §1.6)
CSPs used	All CSPs
Self-tests	SHS / HMAC / AES / DRBG / KDF / Triple-DES / RSA / EC-DH / ECDSA / HW integrity / FW integrity / NDRNG

1.7.2.3 *FIPS mode recommendations*

To use the TPM in a FIPS approved mode of operation (valid for both modes), the TPM operator **shall**:

- Follow the procedure to switch from limited to full operational mode
- Use an encryption session for the commands that inputs/outputs CSPs (List is indicated at §3.2.4.1). For commands without authorization, encryptedSalt used in TPM_StartAuthSession on encryption session creation must be different from the empty buffer.
- Use an approved symmetric algorithm (AES) for encryption sessions

- Use authorization session based on HMAC or policy (no password allowed, cf. §2.2.2.1).
- Duplicate only objects with *encryptedDuplication* attribute set.
- Not use FIPS 140-2 non-approved algorithms:
  - SHA-1 for RSA digital signature generation
  - EC Schnorr for ECC digital signature generation
  - ECDAA for ECC digital signature generation

  For the following services:
  - TPM2_Sign, TPM2_Certify, TPM2_CertifyCreation, TPM2_Quote, TPM2_GetSessionAuditDigest, TPM2_GetCommandAuditDigest, TPM2_GetTime, TPM2_NV_Certify, TPM2_Commit
- Use a policy including TPM2_PolicyAuthValue as a minimum in the policy sequence in case authorization is ensured by policy (authorization by policy must be at least as secure as authorization by HMAC).
- Not use TPM2_LoadExternal service to load TDES keys into the TPM
- Use TPM2_HierarchyChangeAuth after first TPM init or after each TPM2_Clear to set the authorization value for the endorsement, platform, owner and lockout hierarchies.
- Use TPM2_CreatePrimary command only for RSA and ECC key with default template.

If operator does not strictly follow the FIPS approved mode recommendations (ex: use of XOR instead of AES in encryption session), TPM is considered as being in a FIPS non-approved mode of operation.

To use the TPM in a FIPS approved mode if it was previously used in a FIPS non-approved mode, the operator shall:

- Zeroize all data listed in Table 20: Keys and CSPs list that could potentially be reused as CSPs in FIPS approved mode

To use the TPM in a FIPS non-approved mode if it was previously used in a FIPS approved mode, the operator shall:

- Zeroize all CSPs listed in Table 20: Keys and CSPs list that could potentially be used by FIPS non-approved algorithms in FIPS approved mode

1.7.3 *Limited and error modes*

TPM may reach specific states depending on sequence of operation that occurred.

1.7.3.1 ***Shutdown mode***

The shutdown mode is an infinite HW reset loop that may be exit only by a power-off/power-on sequence. This state is entered when TPM detects that a FW integrity check failed during the TPM boot sequence.

1.7.3.2 ***Failure mode***

Failure mode is a state of TPM that restricts the commands that can be executed to TPM_Startup / TPM_GetCapability / TPM_GetTestResult for TPM1.2 and TPM2_GetCapability / TPM2_GetTestResult for TPM2.0. TPM answers to all other commands with a specific error code: TPM_FAILEDSELFTEST (0x1C) for TPM1.2 and TPM_RC_FAILURE (0x101). This state is entered when one (except FW integrity test) of the self-tests fails.

1.7.3.3 ***Reduced mode***

The reduced mode is a specific state of the field upgrade mode (refer to §6) that can be reached if the on-going field upgrade procedure failed due to an error detected in the field upgrade commands received. In reduced mode, only a subset of commands might be executed: TPM_FieldUpgradeStart / TPM_FieldUpgradeData / TPM_GetCapability / TPM_GetTestResult / TPM_ContinueSelfTest for TPM1.2 and TPM2_FieldUpgradeStart / TPM2_FieldUpgradeData / TPM2_GetCapability / TPM2_GetTestResult / TPM2_SelfTest for TPM2.0. TPM answers to all other commands with the error TPM_RC_COMMAND_CODE (0x143). Reduced mode can be exited in case of the reception of a successful TPM_FieldUpgradeStart/TPM2_FieldUpgradeStart command that reloads the previous installed firmware.

1.8 Ports and interfaces

The physical port of the security module is the SPI bus.

The logical interfaces and their mapping to physical ports of the module are described below:

Table 11: Ports and interfaces

Logical interface	Description	Physical port
Control Input Interface	Control Input commands issued to the security module	$\overline{\text{SPI\_CS}}$ / SPI_CLK / MOSI / SPI_RST / PP
Status Output Interface	Status data output by the chip	$\overline{\text{SPI\_CS}}$ / SPI_CLK / MISO / SPI_PIRQ
Data Input Interface	Data provided to the chip as part of the data processing commands	$\overline{\text{SPI\_CS}}$ / SPI_CLK / MOSI
Data Output Interface	Data output by the chip as part of the data processing command	$\overline{\text{SPI\_CS}}$ / SPI_CLK / MISO
Power interface	Power interface of the chip	VPS / GND

Here are some details concerning the ports and interfaces of TPM:

1. The module does not include a maintenance interface.

2. Control and data inputs are multiplexed over the same physical interface. Control and data are distinguished by properly parsing input TPM command parameters according to input structures description, indicated for each TPM2.0 command in **[TPM2.0 Part3 r1.16]**[1] and TPM1.2 command in **[TPM1.2 Part3 r1.16]**[2]**.**

3. Status and data output are multiplexed over the same physical interface. Status and data are distinguished by properly setting output TPM response parameters according to output structures description, indicated for each TPM2.0 command in **[TPM2.0 Part3 r1.16]** and TPM1.2 command in **[TPM1.2 Part3 r1.16].**

4. The logical state machine and the command structure parsing of the module prevent from using input data externally from the "data input path" and prevent from outputting data externally from the "data output path".

5. While performing key generation or key zeroization (no manual key entry on TPM), the output data path is logically disconnected while the output status path remains connected to report any possible failure during command processing. Generally, the output data path is only connected when TPM outputs response containing data.

6. In TPM1.2 mode, plaintext data might be output through usage of:

    - TPM_UnBind
    - TPM_Unseal

    To prevent inadvertent release of the plaintext data, both commands performs:

    - Check of command input structure
    - Check of command authorization (cf. §2.2.1 for details)
    - Decryption of the input blob with private part of specified key

    In TPM2.0 mode, plaintext data might be output through usage of:

    - TPM2_Unseal, TPM2_RSA_Decrypt, TPM2_EncryptDecrypt

    To prevent inadvertent release of the plaintext data, command performs:

    - Check of command input structure
    - Check of command authorization
    - Decryption of the input blob with private part of specified key

    However an encryption session might be used with these commands to avoid releasing plaintext data.

7. The logical state machine and command structure of the module guarantees the inhibition of all data output via the data output interface whenever an error state exists and while doing self-tests.

[1] Some commands only deal with control input and status output parameters

[2] Some commands only deal with control input and status output parameters

2 IDENTIFICATION AND AUTHENTICATION POLICY

This chapter gives details about the roles managed by TPM for TPM1.2 and TPM2.0 modes.

2.1 TPM1.2

2.1.1 Roles

Services (services are listed in §3.2.2) proposed by TPM are accessible under different roles. Next table defines the different roles supported by the TPM.

Table 12: Roles

Role	Description	Type of authentication	Authentication data
Crypto officer (CO)	Equivalent to TPM owner (cf. **[TPM1.2 Part1 r1.16]** for role definition). Some TPM services are reserved to owner (initialization/configuration).	Role based	160-bit secret data (Owner AuthData)
User (U)	Role requiring entity authorization, operator authorization.	Role based	160-bit secret data (key usageAuth or operator AuthData)
Physical presence (PP)	HW assertion that proves that an operator is physically present (no remote access)	HW based	None
No authentication (NA)	Some TPM services do not require any authentication.	None	None

The security module does NOT provide a Maintenance Role or Maintenance Interface.

Cryptographic module does NOT support concurrent operators.

2.1.2 Authentication

2.1.2.1 Description

Crypto officer and user authentication data knowledge must be proven to authorize some TPM services. TPM uses a two-step mechanism for authorization that consists in:

1. Opening a session of the following types:
    a. OIAP: Object-Independent Authorization Protocol
    b. OSAP: Object-Specific Authorization Protocol
    c. DSAP: Delegation-Specific Authorization Protocol

    Session is used to establish a sequence of nonce-data included in the authorization process (protection against replay attacks). OSAP and DSAP sessions also create a shared secret used as HMAC key for command authorization. For OIAP, the authorization data is directly used as HMAC key.
2. Using the command to be authorized by verifying if HMAC (based on authorization value) passed as parameter corresponds to the value computed by TPM. If they match, command execution is authorized.

Secret authorization data is never exposed in plaintext (there is one exception for operatorAuth entered by TPM_SetOperatorAuth service and used by TPM_SetTempDeactivated). HMAC computation output based on the authorization data enables to prove knowledge of this secret.

When power is removed from the module, all existing authentication sessions are destroyed. Therefore, the module must re-authenticate every role or identity after each power-on sequence.

2.1.2.2 *Authorization strength*

As authorization values are 160-bit random values (based on unbiased distribution of ‘0’ and ‘1’), the probability for an attacker to guess the authorization data is:

$$\frac{1}{2^{160}} = 6{,}84 * 10^{-49}$$

This value matches the requirement of $1*10^{-6}$ indicated in **[FIPS 140-2]**.

The number of attempts per minute that an attacker can make is limited by the DAM (Dictionary Attack Mechanism). DAM consists in counting the number of failed authentication. When this counter reaches a pre-defined threshold, a lockout period is started. During this period, no authorized command execution is allowed and a specific error (TPM_DEFEND_LOCK_RUNNING) is returned in TPM response until period expires. Next table indicates the threshold values and the lockout durations:

Table 13: DAM lockout durations

Failed authentication counter	<10	10 (DAM threshold)	11	12	13	…	23	>23
Lockout period (in seconds)	0	10	20	40	60	…	81920	86400

This table indicates that an attacker can do a maximum (during the first minute) of 12 trials per minute (if failed authorization counter reaches 12 it means total lockout period is equal to 10s + 20s + 40s = 70s). As a result the probability per minute that a random attempt will lead to a successful authorization matches FIPS requirements. Value is equal to:

$$12 * \frac{1}{2^{160}} = 8{,}21 * 10^{-48}$$

This value matches the requirement of $1*10^{-5}$ indicated in **[FIPS 140-2]**.

NB: commands handling (reception, processing and response sending) is negligible compared to the lockout periods and not taken into account in the above computation.

2.2 TPM2.0

2.2.1 *Roles*

Services (services are listed in §3.2.2) proposed by TPM are accessible under different roles. Next table defines the different roles supported by the TPM.

Table 14: Roles

Role	Description	Type of authentication	Authentication data
Crypto officer (CO)	Role that requires knowledge of the authValue or authPolicy associated to one of the hierarchy (incl. lockout).	Role based	256-bit secret data (authValue and/or authPolicy)
User (U)	Role that requires knowledge of the authValue or authPolicy associated to one object or NV index.	Role based	160-bit or 256-bit secret data (authValue and/or authPolicy). Authorization depends on userWithAuth object attribute.

Admin (A)	The object Administrator controls the certification of an object (TPM2_Certify and TPM2_ActivateCredential) and controls changing of the authValue of an object (TPM2_ObjectChangeAuth).	Role based	160-bit or 256-bit secret data (authValue and/or authPolicy). Authorization depends on adminWithPolicy object attribute.
DUP (D)	This authorization role is only used for TPM2_Duplicate(). If duplication is allowed, authorization must always be provided by a policy session and the authPolicy equation of the object must contain a command that sets the policy command code to TPM_CC_Duplicate.	Role based	160-bit or 256-bit secret data (authPolicy).

Some commands can also be executed without any authorization role. Those commands are marked as NA in the service list table (Table 21: Command support table).

The security module does NOT provide a Maintenance Role or Maintenance Interface and does NOT support concurrent operators.

Roles are implicitly selected by TPM operator on command execution (cf. Table 21 for correspondence between service and supported role) by proving knowledge of the authorization value or knowledge of the policy sequence (nature of authorization session indicates the type of authorization) that are associated with the object the command is operating on.

An operator might switch from one role to another by executing commands requiring different roles and proving knowledge of the authorization value or policy sequence of objects the role is associated to.

2.2.2 Authentication

2.2.2.1 Description

In FIPS approved mode of operation, TPM uses a mechanism for authorization that consists in:

1. Opening an authorization session that may be of the following types:
    a. HMAC
    b. Policy
2. Executing the expected policy commands sequence in case of policy authorization session (defined policy must follow minimal recommendations listed in §1.7.2.3).
3. Do the comparison between reference value and computed value. If both match, command execution is authorized.

More details on HMAC and policy sessions can be found in §19 of [TPM2.0 Part1 r1.16].

2.2.2.2 Authorization strength

As minimum value of authorization or policy values might be 160-bit random values (based on unbiased distribution of ‘0’ and ‘1’), the probability for an attacker to guess the authorization data is:

$$\frac{1}{2^{160}} = 6.84 * 10^{-49}$$

This value is then higher than the minimum of $1*10^{-6}$ required by **[FIPS140-2]**.

The number of attempts per minute that an attacker can make is limited by the DAM (Dictionary Attack Mechanism). DAM consists in counting the number of failed authentication. When this counter reaches a pre-defined threshold, a lockout period is started. During this period, no authorized command execution is allowed and a specific error is returned in TPM response until period expires. Next table indicates the threshold values and the lockout durations:

Table 15: DAM lockout durations

Failed authentication counter	>31
Lockout period (in seconds)	7200

This table indicates that an attacker can do a maximum (during the first minute) of 32 trials per minute before DAM being active. As a result the probability per minute that a random attempt will lead to a successful authorization matches FIPS requirements. Value is equal to:

$$32 * \frac{1}{2^{160}} = 2.19 * 10^{-47}$$

This value is then higher than the minimum of $1*10^{-5}$ required by **[FIPS140-2]**.

NB: commands handling (reception, processing and response sending) is negligible compared to the lockout periods and not taken into account in the above computation.

NB2: DAM parameters might be changed by using TPM2_DictionnaryAttackParameters command. However to operate in a FIPS approved mode, they shall not be changed in order not to decrease the authorization strength computed above.

2.2.2.3 Authorization protection

By following recommendations to operate in FIPS mode of operation, authorization data associated to objects, NV indexes or hierarchies are never output from TPM in plaintext form and thus are protected from unauthorized disclosure.

Authorization can be changed via the following services:

- TPM2_ObjectChangeAuth
- TPM2_HierarchyChangeAuth
- TPM2_NV_ChangeAuth

As indicated in Table 21, roles that imply authentication are associated with these services meaning that authentication are protected against unauthorized modification and substitution.

TPM authorization mechanism (HMAC or policy digest comparison) does not provide any information about authentication data or policy sequence. Authentication indicates pass (command executed) or fail (command not executed) and does not provide feedback that could weaken the strength of authentication.

3 ACCESS CONTROL POLICY

3.1 TPM1.2

This chapter gives details about the services, keys and CSPs that the TPM manages in TPM1.2 mode.

3.1.1 *List of Keys and CSPs*

Table 16: Keys and CSPs list

Keys/CSPs		Description	Zeroization
Index	**Name**		
1	Endorsement key (EK) – private part	2048-bits permanent RSA key unique per TPM stored in the form of two prime numbers. EK primes are generated externally by a HSM and inserted during TPM production phase. EK is used to: • Decrypt encOwnerAuth and encSrkAuth in TPM_TakeOwnership command • Decrypt blob in TPM_ActivateIdentity	No zeroization (NIST waiver)
2	Storage root key (SRK) – private part & authorization value	2048-bits non-volatile RSA key. Root key of the key storage hierarchy. Key is generated and stored on TPM on TPM_TakeOwnership command according to the input parameters. SRK is used to: • Wrap and unwrap keys stored in the protected storage hierarchy Authorization data (non-volatile data) are 160-bits secret data used for SRK authorization. It is passed encrypted (RSA OAEP SHA1 algorithm with key = public part of EK) to TPM_TakeOwnership command. It is used as key for TPM KDF SP800-135 in session shared secret (CSP #8) generation for TPM_MakeIdentity and might be used for commands with U role in Table 21: Command support table that uses SRK as parent key.	TPM_OwnerClear TPM_ForceClear
3	User RSA keys – private part & authorization value	2048-bits RSA keys generated with TPM_CreateWrapKey, TPM_MakeIdentity and TPM_CMK_CreateKey commands (output encrypted from TPM with parent key indicated in the command). Keys loaded on the TPM via TPM_ActivateIdentity, TPM_LoadKey or TPM_LoadKey2. Depending on key attributes (keyUsage field in TPM_KEY structure), key can be used as: • Signing key (TPM_KEY_SIGNING) • Storage key (TPM_KEY_STORAGE) • Identity key (TPM_KEY_IDENTITY) • Binding key (TPM_KEY_BIND) • Signing and binding key (TPM_KEY_LEGACY) • Migration key (TPM_KEY_MIGRATE) Key might be volatile or non-volatile (keyFlags parameter in TPM_KEY structure).	TPM_OwnerClear TPM_ForceClear TPM_FlushSpecific TPM_EvictKey TPM_Init (for volatile keys only)

Keys/CSPs		Description	Zeroization
Index	Name		
		Authorization data (non-volatile data) are 160-bits secret data used for user RSA key authorization. It is passed encrypted (RSA OAEP SHA1 algorithm with key = public part of parent key) to key creation commands (TPM_CreateWrapKey, TPM_MakeIdentity and TPM_CMK_CreateKey). It is used as key for TPM KDF SP800-135 in session shared secret (CSP #8) generation for commands with U role in Table 21: Command support table that might use user RSA key as parent key.	
4	Field upgrade verification key	2048-bits permanent RSA key unique per TPM product line. Only public part of the key is stored in the TPM (modulus, exponent).	No (public key only)
5	contextKey / delegateKey	128-bits non-volatile AES key used to perform context saves/restores (TPM_SaveContext, TPM_LoadContext) and delegation blobs encryption/decryption (TPM_Delegate_CreateKeyDelegation, TPM_Delegate_CreateOwnerDelegation, TPM_Delegate_LoadOwnerDelegation). Key is generated by HDRBG on TPM_TakeOwnership command.	TPM_OwnerClear TPM_ForceClear
6	HDRBG input seed	48-bytes value output from a NDRNG.	Transient value
8	Session shared secret	160-bit volatile shared secret generated on TPM_OSAP or TPM_DSAP commands execution by derivation (TPM KDF SP800-135) using entity authorization data as key. Session shared secret is used as: • AES CTR key (first 128-bits) in ADIP protocol to encrypt/decrypt authorization data (list of commands is indicated in Table 18: Encrypted methods for secret and private keys input). • HMAC SHA-1 key in HMAC computation in authorization protocols (concerned commands are indicated with CO or U role in Table 21: Command support table).	TPM_FlushSpecific TPM_OwnerClear TPM_ForceClear
9	NV index – authorization value	160-bits (non-volatile data) used as secret authorization data for a specific NV index. Value is passed encrypted (AES CTR 128 with key = OSAP shared secret) to the TPM_NV_DefineSpace command. It is used as key for TPM KDF SP800-135 in session shared secret (CSP #8) generation for TPM_NV_WriteValueAuth and TPM_NV_ReadValueAuth commands.	TPM_OwnerClear TPM_ForceClear
10	HDRBG state	222-bytes (volatile data) representing the HDRBG internal state (V and C secret values). HDRBG is seeded after each reset with NDRNG output. Internal state is updated after each HDRBG generate command execution or reseed. HDRBG is used in random number generation for cryptographic material.	TPM_OwnerClear TPM_ForceClear TPM_SetMode TPM_Init
11	tpmProof	160-bits secret random number (non-volatile data) generated by HDRBG on TPM_TakeOwnership command execution. It is used as: • HMAC SHA-1 key in integrity computation of blobs generated or read in the following commands:	TPM_OwnerClear TPM_ForceClear

Keys/CSPs		Description	Zeroization
Index	Name		
		TPM_CertifyKey2, TPM_Delegate_CreateKeyDelegation, TPM_Delegate_CreateOwnerDelegation, TPM_Delegate_UpdateVerification, TPM_CreateMigrationBlob, TPM_AuthorizeMigrationKey, TPM_CMKApproveMA, TPM_CMK_CreateKey, TPM_CMK_CreateTicket, TPM_CMK_CreateBlob, TPM_CMK_ConvertMigration, TPM_SaveContext, TPM_LoadContext, TPM_Seal	
12	Owner – authorization value	160-bits secret authorization data (non-volatile data) for owner authorization. It is passed encrypted (RSA OAEP SHA1 algorithm with key = public part of EK) to TPM_TakeOwnership command. It can be changed on TPM_ChangeAuthOwner command processing. It is used as key for TPM KDF SP800-135 in session shared secret (CSP #8) generation for all commands listed in Table 21: Command support table and requesting CO role to be authorized.	TPM_OwnerClear TPM_ForceClear
13	Monotonic counters – authorization value	160-bits secret authorization data (non-volatile data) for a specific monotonic counter (up to 4 monotonic counters can be created). Value is passed encrypted (AES CTR 128 with key = OSAP shared secret) to the TPM_CreateCounter command. It is used as key for TPM KDF SP800-135 in session shared secret (CSP #8) generation for TPM_IncrementCounter and TPM_ReleaseCounter commands.	TPM_ReleaseCounter TPM_ReleaseCounter Owner TPM_OwnerClear TPM_ForceClear
14	Pre-computed RSA keys – private part	2048-bits RSA keys (exponent = 65537) pre-computed during TPM background processing (between commands handling) and forming a pool of keys used to speed up key creation commands. Keys are non-volatile data.	TPM_OwnerClear TPM_ForceClear TPM_SetMode
15	Operator – authorization value	160-bits secret authorization data (non-volatile data) entered in plaintext on TPM_SetOperatorAuth. It is used as key for TPM KDF SP800-135 to be able to deactivate the TPM until the next boot of the platform via TPM_SetTempDeactivated command.	TPM_OwnerClear TPM_ForceClear

3.1.2 Services

Next table lists all services supported by the TPM in FIPS approved mode and indicates for each service, the role that can use this service and the keys/CSPs that can be accessed.

Table 17: Command support table

Services		Role	Keys and CSP access (R = read, W = write, O = output, Z = zeroize)	Authorized limited approved mode
Admin Start up and State				
1	TPM_Init	**NA**	W: 1, 11 (first power-up only) Z: 3, 10	X
2	TPM_Startup	**NA**	-	X
3	TPM_SaveState	**NA**	-	X
Admin Testing				
4	TPM_SelfTestFull	**NA**	-	X
5	TPM_ContinueSelfTest	**NA**	-	X
6	TPM_GetTestResult	**NA**	-	X
Admin Opt-in				
7	TPM_SetOwnerInstall	**PP**	-	X
8	TPM_OwnerSetDisable	**CO**	R: 6, 8, 10 W: 6, 10	
9	TPM_PhysicalEnable	**PP**	-	X
10	TPM_PhysicalDisable	**PP**	-	X
11	TPM_PhysicalSetDeactivated	**PP**	-	X
12	TPM_SetTempDeactivated	**U, PP**	R: 6, 10, 15 W: 6, 10	
13	TPM_SetOperatorAuth	**PP**	W: 15	
Admin Ownership				
14	TPM_TakeOwnership	**CO**	R: 1, 6, 8, 10, 12, 14 W: 2, 5, 6, 10, 11, 12	
15	TPM_OwnerClear	**CO**	R: 8, 10, 12 Z: 2, 3, 5, 8, 9, 10, 11, 12, 13, 14, 15	
16	TPM_ForceClear	**PP**	Z: 2, 3, 5, 8, 9, 10, 11, 12, 13, 14, 15	X
17	TPM_DisableOwnerClear	**CO**	R: 8, 10, 12	
18	TPM_DisableForceClear	**NA**	-	X
19	TSC_PhysicalPresence	**NA**	-	X
20	TSC_ResetEstablishmentBit	**NA**	-	X
Capability				
21	TPM_GetCapability	**NA**	O: 4 (SHA-256 of public key)	X
22	TPM_SetCapability	**CO**	R: 6, 8, 10, 12 W: 6, 10	
23	TPM_GetCapabilityOwner	**CO**	R: 6, 8, 10, 12	
Administrative Functions & Management				

Services		Role	Keys and CSP access (R = read, W = write, O = output, Z = zeroize)	Authorized limited approved mode
29	TPM_ResetLockValue	**CO**	R: 3, 6, 8, 10, 12 W: 6, 10	
Storage				
30	TPM_Seal	**U**	R: 3, 6, 8, 10, 11 W: 6, 10	
31	TPM_Unseal	**U**	R: 3, 6, 8, 10, 11 W: 6, 10	
32	TPM_UnBind	**U**	R: 3, 6, 8, 10 W: 6, 10	
33	TPM_CreateWrapKey	**U**	R: 3, 6, 8, 10, 14 W: 6, 10 O: 3 (private part is encrypted)	
34	TPM_LoadKey2	**U**	R: 6, 8, 10, 11 W: 3, 6, 10	
35	TPM_GetPubKey	**U**	R: 3, 6, 8, 10 W: 6, 10	
Migration				
37	TPM_CreateMigrationBlob	**U**	R: 3, 6, 8, 10, 11 W: 6, 10	
38	TPM_ConvertMigrationBlob	**U**	R: 3, 6, 8, 10 W: 6, 10	
39	TPM_AuthorizeMigrationKey	**CO**	R: 3, 6, 8, 10, 11, 12 W: 6, 10	
40	TPM_MigrateKey	**U**	R: 3, 6, 8, 10 W: 6, 10	
41	TPM_CMK_SetRestrictions	**CO**	R: 3, 6, 8, 10, 12 W: 6, 10	
42	TPM_CMK_ApproveMA	**CO**	R: 6, 8, 10, 11, 12 W: 6, 10	
43	TPM_CMK_CreateKey	**U**	R: 2, 3, 6, 8, 10, 11, 14 W: 3, 6, 10 O: 3 (private part is encrypted)	
44	TPM_CMK_CreateTicket	**CO**	R: 2, 3, 6, 8, 10, 12 W: 6, 10	
45	TPM_CMK_CreateBlob	**U**	R: 3, 6, 8, 10, 11 W: 6, 10	
46	TPM_CMK_ConvertMigration	**U**	R: 3, 6, 8, 10, 12 W: 6, 10	
Cryptographic Functions				
52	TPM_SHA1Start	**NA**	**-**	X
53	TPM_SHA1Update	**NA**	**-**	X
54	TPM_SHA1Complete	**NA**	**-**	X
55	TPM_SHA1CompleteExtend	**NA**	**-**	X

Services		Role	Keys and CSP access (R = read, W = write, O = output, Z = zeroize)	Authorized limited approved mode
56	TPM_Sign	U	R: 3, 6, 8, 10 W: 6, 10	
57	TPM_GetRandom	NA	R: 6, 10 W: 6, 10	
58	TPM_StirRandom	NA	R: 6, 10 W: 6, 10	
Endorsement Key Handling				
64	TPM_ReadPubek	NA	-	X
65	TPM_OwnerReadInternalPub	CO	R: 1, 2, 6, 8, 10, 12 W: 6, 10	
Identity Creation and Activation				
66	TPM_MakeIdentity	CO	R: 2, 6, 8, 10, 11, 12, 14 W: 6, 10 O: 3 (identity key, private part is encrypted)	
67	TPM_ActivateIdentity	CO	R: 1, 6, 8, 10, 12 W: 6, 10	
Integrity Collection and reporting				
68	TPM_Extend	NA	-	X
69	TPM_PCRRead	NA	-	X
71	TPM_PCR_Reset	NA	-	X
Changing Auth Data				
73	TPM_ChangeAuth	U	R: 6, 8, 10 W: 3, 6, 9, 10	
74	TPM_ChangeAuthOwner	CO	R: 6, 8, 10, 12 W: 2, 6, 10, 12	
Authorization sessions				
75	TPM_OIAP	NA	R: 6, 10 W: 6, 8, 10	
76	TPM_OSAP	NA	R: 2, 3, 6, 9, 10, 12, 13 W: 6, 8, 10	
77	TPM_DSAP	NA	R: 3, 6, 10, 11 W: 6, 8, 10	
78	TPM_SetOwnerPointer	NA	-	X
Delegation				
79	TPM_Delegate_Manage	CO	R: 6, 8, 10, 12 W: 6, 8, 10	
80	TPM_Delegate_CreateKeyDelegation	U	R: 3, 5, 6, 8, 10, 11, 12 W: 6, 10	
81	TPM_Delegate_CreateOwnerDelegation	CO	R: 5, 6, 8, 10, 11, 12 W: 6, 10	
82	TPM_Delegate_LoadOwnerDelegation	CO	R: 5, 6, 8, 10, 11, 12 W: 6, 10	

Services		Role	Keys and CSP access (R = read, W = write, O = output, Z = zeroize)	Authorized limited approved mode
83	TPM_Delegate_ReadTable	**NA**	-	X
84	TPM_Delegate_UpdateVerification	**CO**	R: 6, 8, 10, 11, 12 W: 6, 10	
85	TPM_Delegate_VerifyDelegation	**NA**	R: 5, 6, 11	
Non-Volatile Storage				
86	TPM_NV_DefineSpace	**CO**	R: 6, 8, 10, 12 W: 6, 9, 10 Z: 9 (if index previously defined and size = 0)	
87	TPM_NV_WriteValue	**CO**	R: 6, 8, 10, 12 W: 6, 10	
88	TPM_NV_WriteValueAuth	**U**	R: 6, 8, 9, 10 W: 6, 10	
89	TPM_NV_ReadValue	**CO**	R: 6, 8, 10, 12 W: 6, 10	
90	TPM_NV_ReadValueAuth	**U**	R: 6, 8, 9, 10 W: 6, 10	
Session Management				
91	TPM_KeyControlOwner	**CO**	R: 6, 8, 10, 12 W: 3, 10	
92	TPM_SaveContext	**NA**	R: 3, 5, 9, 11 Z: 8	
93	TPM_LoadContext	**NA**	R: 3, 5, 9, 11	
Eviction				
94	TPM_FlushSpecific	**NA**	Z: 3, 8	X
Timing Ticks				
95	TPM_GetTicks	**NA**	-	X
Transport Sessions				
97	TPM_EstablishTransport	**U**	R: 3, 6, 8, 10 W: 6, 10	
98	TPM_ExecuteTransport	**U**	R: 6, 8, 10 W: 6, 10	
Monotonic Counter				
100	TPM_CreateCounter	**CO**	R: 6, 8, 10, 12 W: 6, 10, 13	
101	TPM_IncrementCounter	**U**	R: 6, 8, 10, 13 W: 6, 10	
102	TPM_ReadCounter	**NA**	-	X
103	TPM_ReleaseCounter	**U**	R: 6, 8, 10, 13 W: 6, 10 Z: 8, 13	

Services		Role	Keys and CSP access (R = read, W = write, O = output, Z = zeroize)	Authorized limited approved mode
104	TPM_ReleaseCounterOwner	**CO**	R: 6, 8, 10, 12 W: 6, 10 Z: 8, 13	
Signal Commands				
124	TPM_HASH_START	**NA**	-	X
125	TPM_HASH_DATA	**NA**	-	X
126	TPM_HASH_END	**NA**	-	X
Proprietary commands				
127	TPM_FieldUpgradeStart	**CO, PP**	R: 4, 6, 8, 10, 12 W: 6, 10	
128	TPM_FieldUpgradeData	**NA**	-	
129	TPM_SHA256Start	**NA**	-	X
130	TPM_SHA256Update	**NA**	-	X
131	TPM_SHA256Complete	**NA**	-	X
133	TPM_SetMode	**CO**	R: 6, 8, 10, 12 W: 6, 10 Z: 3, 14	
134	TPM_SetCommandSet	**PP**	-	X
Deprecated commands				
134	TPM_EvictKey	**NA**	Z: 3	X
135	TPM_Terminate_Handle	**NA**	-	X
136	TPM_DirWriteAuth	**CO**	R: 6, 8, 10, 11, 12 W: 6, 9, 10	
137	TPM_DirRead	**NA**	R: 9	X
138	TPM_ChangeAuthAsymStart	**U**	R: 3, 6, 8, 10 W: 6, 10	
139	TPM_ChangeAuthAsymFinish	**U**	R: 6, 8, 10 W: 3, 6, 10	
140	TPM_Reset	**NA**	Z: 8	X
141	TPM_OwnerReadPubek	**CO**	R: 1, 6, 8, 10, 12 W: 6, 10	
142	TPM_DisablePubekRead	**CO**	R: 6, 8, 10, 12 W: 6, 10	
143	TPM_LoadKey	**U**	R: 3, 6, 8, 10, 11 W: 6, 10	
Non FIPS service				
144	Field upgrade de-obfuscation[1]	**NA**	**-**	

[1] This service is not callable from TPM interface but is only used internally by TPM_FieldUpgradeData command. It consists of de-obfuscating data received by the TPM_FieldUpgradeData command with a non-FIPS approved algorithm.

3.1.3 *Authorization*

Some of the services listed above manipulate CSPs without requiring the operator to assume an authorized role:

- Services using DRNG (read, state update without manipulation):

  TPM_GetRandom, TPM_StirRandom
- Services used for authentication mechanism:

  TPM_OIAP, TPM_OSAP, TPM_DSAP,
- Services using only public part of objects:

  TPM2_ GetCapability: SHA-256 of public key output.
- Specific services that do not affect security of the module:

  TPM_Delegate_VerifyDelegation: use of CSPs to check blob integrity.

  TPM_SaveContext: use of CSPs to protect blob stored outside of TPM.

  TPM_LoadContext: use of CSPs to check blob integrity.

  TPM_FlushSpecific: flush of data (key, authorization or transport session) stored inside the TPM.

  TPM_FieldUpgradeData: this command can only be executed if TPM_FieldUpgradeStart previously executed (authorization requested).

  TPM_EvictKey: flush of data (key, authorization or transport session) stored inside the TPM.

3.1.4 *Key management*

3.1.4.1 *Key entry and output*

Next table indicates the approved method used to encrypt all secret and private keys (indicated by S for secret value and P for private key in type column), entered into or output from the cryptographic module.

Table 18: Encrypted methods for secret and private keys input

Service	Parameter name	Type	Input or output	Encryption algorithm
TPM_LoadKey	inKey (private part)	P	Input	RSA-OAEP SHA1
TPM_LoadKey2	inKey (private part)	P	Input	RSA-OAEP SHA1
TPM_TakeOwnership	encOwnerAuth encSrkAuth	S S	Input Input	RSA-OAEP SHA1 RSA-OAEP SHA1
TPM_Seal	encAuth	S	Input	AES CTR 128
TPM_CreateWrapKey	dataUsageAuth dataMigrationAuth wrappedKey	S S P	Input Input Output	AES CTR 128 AES CTR 128 RSA-OAEP SHA1
TPM_CMK_CreateKey	dataUsageAuth wrappedKey (private part)	S P	Input Output	AES CTR 128 RSA-OAEP SHA1
TPM_EstablishTransport	secret	S	Input	RSA-OAEP SHA1
TPM_MakeIdentity	identityAuth	S	Input	AES CTR 128
TPM_Delegate_CreateKeyDelegation	delAuth	S	Input	AES CTR 128

TPM_Delegate_CreateOwnerDelegation	delAuth	S	Input	AES CTR 128
TPM_NV_DefineSpace	encAuth	S	Input	AES CTR 128
TPM_CreateCounter	encAuth	S	Input	AES CTR 128
TPM_SaveContext	contextBlob	P	Output	AES CTR 128
TPM_LoadContext	contextBlob	P	Input	AES CTR 128
TPM_CreateMigrationBlob	outData	P	Output	RSA-OAEP SHA1
TPM_ConvertMigrationBlob	inData	P	Input	RSA-OAEP SHA1
TPM_MigrateKey	inData outData	P P	Input Output	RSA-OAEP SHA1 RSA-OAEP SHA1
TPM_CMK_ConvertMigration	outData	P	Output	RSA-OAEP SHA1
TPM_ChangeAuth	encData	S	Input	AES CTR 128

3.1.4.2 Key transport

As indicated in the above table, the TPM supports two different algorithms for key transport. Relative security strength of each cryptographic algorithm supported by the module is indicated in the table below:

Table 19: Cryptographic Functions

Algorithm	Comparable number of bits of security
RSA-2048	112
AES-128[1]	128

RSA-2048 and AES-128 are used to transport RSA-2048 keys (security strength of the transport method is then greater or equal than the security strength of the keys transported).

AES-128 in CTR mode is also used in ADIP protocol to encrypt 160-bits authorization data.

RSA is used with OAEP SHA-1 padding scheme method to encrypt (wrap) and decrypt (unwrap) secrets and private keys, as indicated in Table 18: Encrypted methods for secret and private keys input, with a parent key already loaded into the TPM.

AES is used in CTR mode to encrypt/decrypt with shared secret from OSAP session as key for all commands listed in Table 18: Encrypted methods for secret and private keys input except for TPM_SaveContext and TPM_LoadContext that uses contextKey.

[1] AES is used in conjunction with HMAC-SHA-1 approved authentication method (scheme is compliant with **[SP800-38F]**)

3.2 TPM2.0

This chapter gives details about the services, keys and CSPs that the TPM manages in TPM2.0 mode.

3.2.1 List of Keys and CSPs

Table 20: Keys and CSPs list

<table>
<tr><th colspan="2">Keys/CSPs</th><th rowspan="2">Description</th><th rowspan="2">Zeroized</th></tr>
<tr><th>Index</th><th>Name</th></tr>
<tr><th colspan="4">Hierarchies</th></tr>
<tr><td>1</td><td>nullSeed</td><td rowspan="4">32 bytes primary seed values resp. for NULL, platform, endorsement and storage hierarchies.
NullSeed is a random value generated by HDRBG at each TPM power-up.
PpSeed / epSeed / spSeed are random values generated by HDRBG at first TPM power-up.
They are used as keys for:
KDFa to derive seedValue during object creation (cf. [TPM2.0 Part1 r1.16] §27.6.4)
KDFa to generate a symmetric encryption key used in TPM2B_PRIVATE structure en/decryption.
KDFa to generate HMAC key used in TPM2B_PRIVATE integrity protection generation or verification
They are used as seeds for:
DRBG to generate random as input to prime numbers (RSA) and private key generation (ECC)</td><td>TPM reset</td></tr>
<tr><td>2</td><td>ppSeed</td><td>TPM2_Change PPS</td></tr>
<tr><td>3</td><td>epSeed</td><td>TPM2_Change EPS</td></tr>
<tr><td>4</td><td>spSeed</td><td>TPM2_Clear</td></tr>
<tr><td>5</td><td>nullProof</td><td rowspan="4">32 bytes secret values resp. for NULL, platform, endorsement and storage hierarchies.
NullProof is a random value generated by HDRBG at each TPM power-up.
PhProof / ehProof / shProof are random values generated by HDRBG at first TPM power-up.
They are used as keys for:
KDFa to generate context encryption key and IV (cf. [TPM2.0 Part1 r1.16] §30.3.1)
HMAC to compute context blob integrity (cf. [TPM2.0 Part1 r1.16] §30.3.2)
HMAC to compute/verify tickets
shProof is used also as key for:
KDFa to generate obfuscation value used in attestation commands (cf. [TPM2.0 Part1 r1.16] §36.7)</td><td>TPM reset</td></tr>
<tr><td>6</td><td>phProof</td><td>TPM2_Change PPS</td></tr>
<tr><td>7</td><td>ehProof</td><td>TPM2_Clear</td></tr>
<tr><td>8</td><td>shProof</td><td>TPM2_Clear</td></tr>
<tr><td>9</td><td>platformAuth</td><td rowspan="4">32 bytes authorization data (authValue) used in authorization session based resp. on platform, endorsement, and storage or lockout hierarchy authorization.
PlatformAuth is set to 0 at each TPM2_Startup (CLEAR).
EndorsementAuth / ownerAuth / lockoutAuth are set to 0 at first TPM power-up.
Primary auth values can be changed with command TPM2_HierarchyChangeAuth.
They are used as keys for:
HMAC authorization in case of unsalted and unbound session
KDFa to generate session key used in HMAC authorization in case of bound session
They are used as part of keys for:
HMAC authorization in case of salted or bound session (key is concatenation of sessionKey and authValue)
KDFa to generate session key used in HMAC authorization in case of salted and bound session (key is concatenation of authValue and salt)</td><td>TPM2_Startup</td></tr>
<tr><td>10</td><td>endorsementAuth</td><td>TPM2_Clear / TPM2_Change EPS</td></tr>
<tr><td>11</td><td>ownerAuth</td><td>TPM2_Clear</td></tr>
<tr><td>12</td><td>lockoutAuth</td><td>TPM2_Clear</td></tr>
</table>

		They are used as reference values for comparison in case of password authorization session.	
13	platformPolicy	32 bytes authorization data (authPolicy) used in authorization session based resp. on platform, endorsement, storage or lockout hierarchy policy. platformPolicy is set to 0 at each TPM2_Startup (CLEAR). endorsementPolicy / ownerPolicy / lockoutPolicy are set to 0 at first TPM power-up. Primary policies can be changed with command TPM2_SetPrimaryPolicy. They are used as reference values for a comparison in case of policy session.	TPM2_Change PPS
14	endorsementPolicy		TPM2_Clear / TPM2_Change EPS
15	ownerPolicy		TPM2_Clear
16	lockoutPolicy		TPM2_Clear
Objects			
17	authValue	0 to 32 bytes authorization data defined during object creation (TPM2_Create/TPM2_CreatePrimary) used to authorize commands based on this object. Value can be changed with command TPM2_ObjectChangeAuth. It is used as: • HMAC and/or KDFa keys or part of keys in authorization session based on HMAC or password (usage is the same than for CSPs 9/10/11/12)	TPM2_Clear (owner & endorsement) TPM2_Change PPS (platform) TPM2_Change EPS (endorsement)
18	authPolicy	0 to 32 bytes authorization data defined during object creation (TPM2_Create/TPM2_CreatePrimary) used to authorize commands based on this object. It is used as reference value for a comparison in case of policy session	TPM2_Clear (owner & endorsement) TPM2_Change PPS (platform) TPM2_Change EPS (endorsement)
19	seed	32 random bytes generated by HDRBG if object parent is not a hierarchy (if hierarchy, primary seed is used, cf. CSPs 1/2/3/4). It is used as key for: • KDFa to generate seedValue and sensitive during object creation (cf. [TPM2.0 Part1 r1.16] §27.6.4)	Transient value only available during object creation
20	seedValue	32 bytes generated from seed (CSP 19) or one of the primary seeds (CSP 1/2/3/4) through use of KDFa. Set to 0 for asymmetric keys that are not used as storage key. It is used (when not set to 0) as: • Data in SHA computation to generate object's unique value (HMAC and symmetric key creation) • Key in KDFa to generate a symmetric encryption key used in TPM2B_PRIVATE structure en/decryption. • Key in KDFa to generate HMAC key used in TPM2B_PRIVATE integrity protection generation or verification	TPM2_Clear (owner & endorsement) TPM2_Change PPS (platform) TPM2_Change EPS (endorsement)
21	symKey	16 bytes generated from derivation of seedValue through KDFa usage. It is used as key for: • Symmetric en/decryption of TPM2B_PRIVATE structure	Transient value only available during command processing
22	hmacKey	32 bytes generated from derivation of seedValue through KDFa usage. It is used as key for: • HMAC used in TPM2B_PRIVATE integrity protection generation or verification	Transient value only available during command processing

23	sensitive	Object sensitive part that might be passed as encrypted parameter to TPM2_Create command or generated with KDFa from seed or primary seed (if sensitiveDataOrigin attribute is set) in case object is a symmetric key or a HMAC key. For RSA or ECC key, sensitive corresponds to the private key. Depending on object's nature, sensitive is used as key for: • en/decryption (RSA, AES, TDES) • signature generation (RSA, ECDSA, HMAC) • secret value exchange (ECDH) Available key lengths correspond to the ones listed in **Table 4: Approved algorithms** (Key nature and length are selected by user thanks to the interface of the keys creation commands).	TPM2_Clear (owner & endorsement) TPM2_Change PPS (platform) TPM2_Change EPS (endorsement)
NV Indexes			
24	authValue	0 to 32 bytes authorization data defined during NV index creation (TPM2_DefineSpace) used to authorize commands based on this object. Value can be changed with command TPM2_NV_ChangeAuth. It is used as: • HMAC and/or KDFa keys or part of keys in authorization session based on HMAC or password. • Secret value extended into policyDigest on TPM2_PolicySecret command	TPM2_NV_Und efineSpace / TPM2_NV_Und efineSpaceSpec ial
25	authPolicy	0 to 32 bytes authorization data defined during object creation (TPM2_DefineSpace) used to authorize commands based on this object. It is used as reference value for a comparison in case of policy session	TPM2_NV_Und efineSpace / TPM2_NV_Und efineSpaceSpec ial
Sessions			
26	salt	Value passed encrypted (with a loaded decrypt key) to TPM2_StartAuthSession. It is used as: • Part of KDFa key to generate the sessionKey (cf. [TPM2.0 Part1 r1.16] §19.6)	Transient value only available during TPM2_StartAut hSession processing
27	sessionKey	Key generated by KDFa (cf. [TPM2.0 Part1 r1.16] §19.6) and whose value depends on salt and bind parameters of TPM2_StartAuthSession command (size depends on symmetric algorithm used). It is used as: • HMAC key used to generate and verify command authorization • Part of KDFa key used to generate encryption key and IV of encryption-based session	TPM2_FlushCo ntext
28	encryption key and IV of encryption-based session	Symmetric key and IV generated by KDFa (cf. [TPM2.0 Part1 r1.16] §21.3) from sessionKey and object's authValue. It is used as key and IV for: • Symmetric en/decryption of first parameter of command/response if parameter structure is of type TPM2B_	TPM2_FlushCo ntext
Context			
29	contextKey	16 bytes randomly generated by HDRBG at each TPM reset. It is used as: • Key in KDFa to generate a symmetric encryption key and IV used in context blob en/decryption	TPM reset
30	symKey, IV	2*16 bytes derived from contextKey. It is used as key and IV for: • Symmetric en/decryption of context blob	Transient value only available during TPM2_ContextS ave / TPM2_ContextL oad processing

Duplication			
31	inner symKey	Symmetric key passed as input to duplication commands or generated by HDRBG (size depends on symmetric algorithm used). It is used as: • Symmetric en/decryption key to protect TPM2B_PRIVATE output structure	Transient value only available during command processing
32	seed	32 bytes value randomly generated by HDRBG if new parent is a RSA key or via KDFe if new parent is an ECC key. It is used as key for : • KDFa to generate a symmetric en/decryption key for outer protection • KDFa to generate a HMAC key for outer integrity protection	
33	outer symKey	Symmetric key generated via KDFa from seed. It is used as key for: • Symmetric en/decryption key for outer protection of TPM2B_PRIVATE output structure	
34	outer hmacKey	HMAC key generated via KDFa from seed. It is used as key for: • HMAC integrity key for outer protection of TPM2B_PRIVATE output structure	
Credential			
35	seed	32 bytes value randomly generated by HDRBG if new parent is a RSA key or via KDFe if new parent is an ECC key. It is used as key for : • KDFa to generate a symmetric en/decryption key for outer protection • KDFa to generate a HMAC key for outer integrity protection	Transient value only available during command processing
36	symKey	Symmetric key generated via KDFa from seed. It is used as key for: • Symmetric en/decryption key for outer protection of credentialBlob	
37	hmacKey	HMAC key generated via KDFa from seed. It is used as key for: • HMAC integrity key for outer protection of credentialBlob	
DRBG			
38	DRBG state	Internal state (V and C secret values) of the HDRBG (based on SHA256) stored in RAM.	TPM2_Clear
ECC			
39	commitNonce	32 bytes value randomly generated by HDRBG at each TPM2_Startup (CLEAR). It is used as key for : • KDFa to generate an ECC ephemeral private key used in TPM2_EC_Ephemeral command / TPM2_ZGen_2Phase	Transient value only available during command processing
40	ephemeral key – derived from commitNonce	ECC private key (size depends on curve selected) generated with KDFa from commitNonce. It is used as ephemeral private key in: • TPM2_Ephemeral command (scalar multiplication) to generate the associated ephemeral public key • TPM2_ZGen_2Phase (ECDH scheme) to generate outZ2 (output point)	
41	ephemeral key	ECC private key (size depends on curve selected) generated with HDRBG. It is used as ephemeral private key in: • TPM2_ECDH_KeyGen command (ECDH scheme) to generate zPoint (output point)	
Static keys			
42	Endorsement key - RSA primes	2 primes of 1024 bits used to construct EK if parameters in TPM2_CreatePrimary command match the default EK RSA template. Generated by FIPS140-2 compliant HSM.	TPM2_Change EPS

43	Endorsement key - ECC private key	ECC 256 bits private key used to construct EK if parameters in TPM2_CreatePrimary command match the default EK ECC template. Generated by FIPS140-2 compliant HSM.	TPM2_Change EPS
Field upgrade keys			
44	Field upgrade verification key	2048 bits permanent RSA key unique per TPM product line. Only public part of the key is stored in the TPM (modulus, exponent).	No (public key)
Transient DRBG			
47	Transient DRBG state	Internal state (V and C secret values) of a HDRBG instance (based on SHA256) stored in RAM. HDRBG is instantiated from primary seeds and used only in TPM2_CreatePrimary to generate prime numbers for primary RSA keys.	Transient DRBG state cleared at the end of random numbers generation
DRBG input seed			
48	DRBG input seed	48-bytes value output from a NDRNG.	Transient value

3.2.2 Services

Next table lists all services supported by the TPM and indicates for each service, the role that can use this service and the keys/CSPs that can be accessed.

Table 21: Command support table

Services		**Role**	**Keys and CSP access** W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	**Authorized in limited approved mode**
Start-up				
1	_TPM_Init	**NA**	W (first boot only) : 2, 3, 4, 6, 7, 8, 10, 11, 12, 14, 15, 16 W : 29, 38, 48	X
2	TPM2_Startup	**NA**	W : 1, 5, 9, 13, 39	X
3	TPM2_Shutdown	**NA**	**-**	X
Testing				
4	TPM2_SelfTest	**NA**	**-**	X
5	TPM2_IncrementalSelfTest	**NA**	**-**	X
6	TPM2_GetTestResult	**NA**	**-**	X
Session commands				
7	TPM2_StartAuthSession	**NA**	W : 26, 27, 38, 48 C : 9, 10, 11, 12, 17, 24, 26, 28, 38, 48	
8	TPM2_PolicyRestart	**NA**	-	
Objects commands				
9	TPM2_Create	**U**	R : 18 W : 19, 20, 21, 22, 23, 28, 38, 48 C : 17, 19, 20, 21, 22, 27, 28, 38, 48 O : 20, 23	
10	TPM2_Load	**U**	R : 18 W : 17, 18, 20, 21, 22, 23, 28, 38, 48 C : 17, 19, 20, 21, 22, 27, 28, 38, 48	

Services		Role	Keys and CSP access W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	Authorized in limited approved mode
11	TPM2_LoadExternal	**NA**	W : 17, 18, 20, 21, 22, 23, 28, 38, 48 C : 17, 19, 20, 21, 22, 27, 28, 38, 48	X
12	TPM2_ReadPublic	**NA**	R : 23 W : 28 C : 28	X
13	TPM2_ActivateCredential	**A, U**	R : 18, 23, 35 W : 28, 36, 37, 38, 48 C : 27, 28, 35, 36, 37, 38, 48	
14	TPM2_MakeCredential	**NA**	R : 23 W : 28, 35, 36, 37 C : 28, 36, 37 O : 35	
15	TPM2_Unseal	**U**	R : 18, 23 W : 28, 38, 48 C : 27, 28, 38, 48 O : 23	
16	TPM2_ObjectChangeAuth	**A**	R : 18 W : 17, 28, 38, 48 C : 27, 28, 38, 48	
Duplication commands				
17	TPM2_Duplicate	**D**	R : 18 W : 28, 31, 32, 33, 34, 38, 48 C : 27, 28, 31, 32, 33, 34, 38, 48 O : 23, 31, 32	
18	TPM2_Rewrap	**U**	R : 18, 32 W : 28, 31, 32, 33, 34, 38, 48 C : 27, 28, 31, 32, 33, 34, 38, 48 O : 23, 31, 32	
19	TPM2_Import	**U**	R : 18, 32 W : 28, 31, 33, 34, 38, 48 C : 27, 28, 31, 32, 33, 34, 38, 48 O : 23	
Asymmetric primitives				
20	TPM2_RSA_Encrypt	**NA**	C: 28	
21	TPM2_RSA_Decrypt	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
22	TPM2_ECDH_KeyGen	**NA**	W : 28, 41 C : 28, 41	
23	TPM2_ECDH_ZGen	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
24	TPM2_ECC_Parameters	**NA**	-	X

Services		Role	Keys and CSP access W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	Authorized in limited approved mode
25	TPM2_ZGen_2Phase	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 40, 48	
Symmetric primitives				
26	TPM2_EncryptDecrypt	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
27	TPM2_Hash	**NA**	W: 28 C: 28	
28	TPM2_HMAC	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
Random number generator				
29	TPM2_GetRandom	**NA**	C : 28, 38, 48	X
30	TPM2_StirRandom	**NA**	W : 28, 38, 48 C: 28	X
Hash/HMAC/Event sequences				
31	TPM2_HMAC_Start	**U**	R : 18 W : 17, 28, 38, 48 C : 23, 27, 28, 38, 48	
32	TPM2_HashSequenceStart	**NA**	W: 17, 28 C: 28	X
33	TPM2_SequenceUpdate	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
34	TPM2_SequenceComplete	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
35	TPM2_EventSequenceComplete	**U**	R : 18 W : 28, 38, 48 C : 23, 27, 28, 38, 48	
Attestation commands				
36	TPM2_Certify	**A, U**	R : 18 W : 28, 38, 48 C : 8, 23, 27, 28, 38, 48	
37	TPM2_CertifyCreation	**U**	R : 18 W : 28, 38, 48 C : 8, 23, 27, 28, 38, 48	
38	TPM2_Quote	**U**	R : 18 W : 28, 38, 48 C : 8, 23, 27, 28, 38, 48	
39	TPM2_GetSessionAuditDigest	**CO**	R : 18 W : 28, 38, 48 C : 8, 23, 27, 28, 38, 48	

Services		Role	**Keys and CSP access** W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	**Authorized in limited approved mode**
40	TPM2_GetCommandAuditDigest	**CO**	R : 18 W : 28, 38, 48 C : 8, 23, 27, 28, 38, 48	
41	TPM2_GetTime	**CO**	R : 18 W : 28, 38, 48 C : 8, 23, 27, 28, 38, 48	
Ephemeral EC keys				
43	TPM2_EC_Ephemeral	**NA**	W : 28, 40 C : 28, 39	
Signing and signature verification				
44	TPM2_VerifySignature	**NA**	R : 23 W : 28 C : 5, 6, 7, 8, 28	
45	TPM2_Sign	**U**	R : 18 W : 28, 38, 48 C : 5, 6, 7, 8, 23, 27, 28, 38, 48	
Command audit				
46	TPM2_SetCommandCodeAuditStatus	**CO**	R : 13, 18 C : 9, 11, 15, 27	
Integrity collection (PCR)				
47	TPM2_PCR_Extend	**U**	R : 18 C : 27	
48	TPM2_PCR_Event	**U**	R : 18 W : 28, 38, 48 C : 27, 28, 38	
49	TPM2_PCR_Read	**NA**	-	X
50	TPM2_PCR_Allocate	**CO**	R : 13, 18 C : 9, 27	
53	TPM2_PCR_Reset	**NA**	-	
54	_TPM_Hash_Start	**NA**	-	X
55	_TPM_Hash_Data	**NA**	-	X
56	_TPM_Hash_End	**NA**	-	X
Enhanced authorization commands				
57	TPM2_PolicySigned	**NA**	C: 28	
58	TPM2_PolicySecret	**U**	R : 18 W : 28, 38, 48 C : 9, 10, 11, 12, 17, 24, 27, 28, 38, 48	
59	TPM2_PolicyTicket	**NA**	W : 28 C: 28	
60	TPM2_PolicyOR	**NA**	-	
61	TPM2_PolicyPCR	**NA**	W : 28 C: 28	
62	TPM2_PolicyLocality	**NA**	-	

Services		Role	**Keys and CSP access** W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	Authorized in limited approved mode
63	TPM2_PolicyNV	**U**	R : 18 W : 28 C : 27, 28	
64	TPM2_PolicyCounterTimer	**NA**	W : 28 C: 28	
65	TPM2_PolicyCommandCode	**NA**	-	
66	TPM2_PolicyPhysicalPresence	**NA**	-	
67	TPM2_PolicyCpHash	**NA**	W : 28 C: 28	
68	TPM2_PolicyNameHash	**NA**	W : 28 C: 28	
69	TPM2_PolicyDuplicationSelect	**NA**	W : 28 C: 28	
70	TPM2_PolicyAuthorize	**NA**	W : 28 C: 28	
71	TPM2_PolicyAuthValue	**NA**	-	
72	TPM2_PolicyPassword	**NA**	-	
73	TPM2_PolicyGetDigest	**NA**	W : 28 C: 28	
74	TPM2_PolicyNvWritten	**NA**	-	
Hierarchy commands				
75	TPM2_CreatePrimary	**CO**	R : 13, 14, 15, 16, 18, 42, 43 W : 20, 21, 22, 23, 28, 38, 47, 48 C : 1, 2, 3, 4, 17, 19, 20, 21, 22, 27, 28, 38, 42, 43, 48 Z : 47	
76	TPM2_HierarchyControl	**CO**	C : 9, 10, 11, 27	
77	TPM2_SetPrimaryPolicy	**CO**	W : 13, 14, 15, 16, 28 C : 9, 10, 11, 12, 27, 28	
78	TPM2_ChangePPS	**CO**	Z : 2, 6, 13,14, 17, 18, 20, 23, 43	
79	TPM2_ChangeEPS	**CO**	Z : 3, 7, 10, 14, 17, 18, 20, 23, 42	
80	TPM2_Clear	**CO**	R : 13, 16 W : 38, 48 Z : 4, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 20, 23, 24, 25, 38, 48 C : 38	
81	TPM2_ClearControl	**CO**	R : 13, 16 W : 38, 48 C : 9, 12, 38, 48	
82	TPM2_HierarchyChangeAuth	**CO**	R : 13, 16 W : 9, 10, 11, 12, 28, 38, 48 C : 9, 10, 11, 12, 28, 38, 48	
Non-Volatile Storage				
83	TPM2_DictionaryAttackLockReset	**CO**	R : 16 W : 38, 48 C : 12, 38, 48	
84	TPM2_DictionaryAttackParameters	**CO**	R : 16 W : 38, 48 C : 12, 38, 48	

Services		Role	**Keys and CSP access** W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	Authorized in limited approved mode
Field Upgrade				
86	TPM2_FieldUpgradeStart	**CO**	W : 28 C : 9, 13, 28, 44	
87	TPM2_FieldUpgradeData	**NA**	-	
Context Management				
88	TPM2_ContextSave	**NA**	W : 30 C : 29, 30	
89	TPM2_ContextLoad	**NA**	W : 30 C : 29, 30	
90	TPM2_FlushContext	**NA**	Z : 17, 18, 20, 23, 27, 28	X
91	TPM2_EvictControl	**CO**	R : 13, 15 C : 9, 11	
Clock and Timers				
92	TPM2_ReadClock	**NA**	-	X
93	TPM2_ClockSet	**CO**	R : 13, 15 C : 9, 11	
94	TPM2_ClockRateAdjust	**CO**	R : 13, 15 C : 9, 11	
Capability Commands				
95	TPM2_GetCapability	**NA**	-	X
96	TPM2_TestParms	**NA**	-	X
Non-volatile storage				
97	TPM2_NV_DefineSpace	**CO**	R : 13, 15, 18 W : 24, 25, 28, 38, 48 C : 9, 11, 27, 28, 38, 48	
98	TPM2_NV_UndefineSpace	**CO**	W : 38, 48 C : 27, 38, 48 Z : 24, 25	
99	TPM2_NV_UndefineSpaceSpecial	**CO, A**	W : 38, 48 C : 27, 38, 48 Z : 24, 25	
100	TPM2_NV_ReadPublic	**NA**	C: 28	X
101	TPM2_NV_Write	**U**	W : 28, 38, 48 R : 25 C : 27, 28, 38, 48	
102	TPM2_NV_Increment	**U**	W : 38, 48 R : 25 C : 27, 38, 48	
103	TPM2_NV_Extend	**U**	W : 28, 38, 48 R : 25 C : 27, 28, 38, 48	

Services		Role	Keys and CSP access W = write, O = output, Z = zeroize C = used as key in cryptographic operation R = read (and not used as C)	Authorized in limited approved mode
104	TPM2_NV_SetBits	**U**	W : 38, 48 R : 25 C : 27, 38, 48	
105	TPM2_NV_WriteLock	**U**	W : 38, 48 R : 25 C : 27, 38, 48	
106	TPM2_NV_GlobalWriteLock	**CO**	W : 38, 48 C : 27, 38, 48	
107	TPM2_NV_Read	**U**	W : 28, 38, 48 R : 25 C : 27, 28, 38, 48	
108	TPM2_NV_ReadLock	**U**	W : 38, 48 R : 25 C : 27, 38, 48	
109	TPM2_NV_ChangeAuth	**A**	W : 24, 28, 38, 48 C : 27, 28, 38, 48	
110	TPM2_NV_Certify	**U**	W : 28, 38, 48 R : 25 C : 27, 28, 38, 48	
Proprietary commands				
111	TPM2_SetMode	**CO**	W : 28, 38, 48 C : 27, 28, 38, 48	
112	TPM2_SetCommandSet	**CO**	W : 28, 38, 48 C : 27, 28, 38, 48	
113	TPM2_RestoreEK	**CO**	Z : 3, 7, 10, 14, 17, 18, 20 R : 42, 43 W : 23, 28, 38, 48 C : 27, 28, 38, 48	
114	TPM2_SetCommandSetLock	**CO**	W : 28, 38, 48 C : 27, 28, 38, 48	
Misc commands				
115	TPM2_PP_Commands	**CO**	-	
Non FIPS services				
116	Field upgrade de-obfuscation[1]	**NA**	-	

3.2.3 Authorization

Some of the services listed above manipulate CSPs without requiring the operator to assume an authorized role:

- Services restricted to use of SHS:

[1] This service is not callable from TPM interface but is only used internally by TPM2_FieldUpgradeData command. It consists of de-obfuscating data received by the TPM2_FieldUpgradeData command with a non-FIPS approved algorithm.

TPM2_Hash,	TPM2_HashSequenceStart	

- Services using DRNG (read, state update without manipulation):

TPM2_GetRandom,	TPM2_StirRandom	

- Services used for authentication mechanism:

TPM2_StartAuthSession,	TPM2_PolicySigned,	TPM2_PolicyTicket,
TPM2_PolicyPCR,	TPM2_PolicyCounterTimer	TPM2_PolicyLocality,
TPM2_PolicyCpHash,	TPM2_PolicyNameHash,	TPM2_PolicyAuthorize,
TPM2_PolicyAuthorize,	TPM2_PolicyDuplicationSelect,	TPM2_PolicyGetDigest

- Services using (read, cryptographic operation) only public part of objects:

TPM2_ReadPublic,	TPM2_RSA_Encrypt,	TPM2_NV_ReadPublic

- Specific services that do not affect security of the module:

  TPM2_LoadExternal: loaded object not considered as protected object (specific attribute).

  TPM2_MakeCredential: convenience function that do not use TPM secrets.

  TPM2_ECDH_KeyGen: ephemeral ECC key generation

  TPM2_EC_Ephemeral: ephemeral ECC key generation

  TPM2_FieldUpgradeData: transport command for field upgrade. Can be used only if TPM2_FieldUpgradeStart command has been successfully executed (authorized command)

  TPM2_ContextSave: save objects under an encrypted and integrity protected format

  TPM2_ContextLoad: load encrypted and integrity protected objects into TPM

  TPM2_FlushContext: flush loaded object/session from TPM volatile memory

3.2.4 Key management

3.2.4.1 Key entry and output

Next table indicates the approved method used to encrypt all secret, private keys and data (indicated by S for secret value, P for private key and D for user defined data in type column), entered into or output from the cryptographic module.

Table 22: Encrypted methods for secret and private keys input

Service	Parameter name	Type	Input or output	Encryption algorithm
TPM2_ActivateCredential	credentialBlob	S	Input	AES CFB
	secret	S	Input	RSA OAEP or ECDH
TPM2_ContextSave	context	D	Output	AES CFB
TPM2_ContextLoad	context	D	Input	AES CFB
TPM2_Create	inSensitive	P / S	Input	AES CFB (*)
	outPrivate	P / S	Output	AES CFB
TPM2_CreatePrimary	inSensitive	P / S	Input	AES CFB (*)
TPM2_Duplicate	encryptionKey (if present)	S	Input	AES CFB (*)
	encryptionKeyOut	S	Output	AES CFB (*)
	duplicate	S	Output	AES CFB
	outSymSeed	S	Output	RSA OAEP or ECDH
TPM2_EventSequenceComplete	buffer	D	Input	AES CFB (*)
TPM2_GetRandom	randomBytes	D	Output	AES CFB (**)
TPM2_Hash	data	D	Input	AES CFB (*)
TPM2_HashSequenceStart	auth	S	Input	AES CFB (*)
TPM2_HierarchyChangeAuth	newAuth	S	Input	AES CFB (*)
TPM2_HMAC	buffer	D	Input	AES CFB (*)
TPM2_HMACStart	auth	S	Input	AES CFB (*)
TPM2_Import	encryptionKey (if present)	S	Input	AES CFB (*)
	duplicate	S	Input	AES CFB
	inSymSeed	S	Input	RSA OAEP or ECDH
	outPrivate	S	Output	AES CFB
TPM2_Load	inPrivate	P / S	Input	AES CFB
TPM2_LoadExternal	inPrivate	P / S	Input	AES CFB (*)
TPM2_MakeCredential	credentialBlob	S	Output	AES CFB
	secret	S	Output	RSA OAEP or ECDH
TPM2_NV_ChangeAuth	newAuth	S	Input	AES CFB (*)
TPM2_NV_DefineSpace	auth	S	Input	AES CFB (*)
TPM2_NV_Extend	data	D	Input	AES CFB (*)

TPM2_NV_Read	data	D	Output	AES CFB (**)
TPM2_NV_Write	data	D	Input	AES CFB (*)
TPM2_ObjectChangeAuth	newAuth	S	Input	AES CFB (*)
	outPrivate	S	Output	AES CFB
TPM2_PCR_Event	eventData	D	Input	AES CFB (*)
TPM2_Rewrap	inDuplicate	S	Input	AES CFB
	inSymSeed	S	Input	RSA OAEP or ECDH
	outDuplicate	S	Output	AES CFB
	outSymSeed	S	Output	RSA OAEP or ECDH
TPM2_RSA_Decrypt	message	D	Output	AES CFB (**)
TPM2_RSA_Encrypt	message	D	Input	AES CFB (*)
TPM2_SequenceComplete	buffer	D	Input	AES CFB (*)
TPM2_SequenceUpdate	buffer	D	Input	AES CFB (*)
TPM2_SetPrimaryPolicy	authPolicy	S	Input	AES CFB (*)
TPM2_StirRandom	inData	D	Input	AES CFB (*)
TPM2_Unseal	outData	D	Output	AES CFB (**)
TPM2_EncryptDecrypt	outData	D	Output	AES CFB (**)

(*): Parameter decryption is ensured by use of a decryption session (attribute DECRYPT set)

(**): Parameter encryption is ensured by use of an encryption session (attribute ENCRYPT set). This is mandatory for TPM_Unseal if output data might be used next used as a CSP.

3.2.4.2 *Key transport*

Relative security strength has been calculated for each cryptographic algorithm supported by the module and used for key transport.

Table 23: Cryptographic Functions

Algorithm	Comparable number of bits of security
RSA OAEP (2048 bits)	112
ECDH (P-224 curve)	112
ECDH (P-256 curve)	128
AES CFB (128 bits)[1]	128
AES CFB (256 bits)[2]	256

[1] AES is used in conjunction with HMAC approved authentication method (**[SP800-38F]** compliant)
[2] AES is used in conjunction with HMAC approved authentication method (**[SP800-38F]** compliant)

4 SELF-TESTS

4.1 TPM1.2

Self-tests run by the cryptographic module are split in three categories:

- Power-up self-tests
- Full self-tests
- Conditional self-tests

The power-on self-tests do not require operator intervention in order to run. Power-on self-tests execution completes all tests listed in Table 7: TPM1.2 limited approved mode. Completion of power-on self-tests allows the TPM to be in a limited approved mode allowing to process only a subset of TPM commands (see §1.7.1.1).

To switch from limited approved mode to full approved mode, operator shall execute TPM_SelfTestFull command. This command requests the module to switch mode by executing all self-tests listed in Table 29: Asymmetric cryptography self-tests list (power-up self-tests plus the remaining self-tests, that mainly concern asymmetric cryptography).

The security module outputs an “error” Return Code via the status interface when the error state is entered due to a failed self-test. While in error state, security module does not perform any cryptographic functions and all data output via the data output interface are inhibited.

If power-on self-tests have passed successfully, no status is indicated but commands that require self-tests to be completed can be successfully executed.

4.1.1 Power-up tests list

Table 24: Cryptographic algorithm KATs

Algorithm tested	Test description
SHA1	SHA1 computation on known data (16 bytes) and comparison of output to the expected digest (20 bytes)
SHA256	SHA256 computation on known data (16 bytes) and comparison of output to the expected digest (32 bytes)
NDRNG	TPM performs AIS31 statistical test verification on NDRNG output. If test fails, status is set to FAIL and error is returned.

Table 25: TPM integrity tests

Algorithm tested	Test description
FW integrity	FW integrity is verified by computing an EDC (CRC-16 ISO 13239) and comparing it to reference values. FW integrity is verified during boot sequence before execution of one of the code block (CML, AFL and TPM) and during full self-tests execution. If failure is detected during boot sequence, TPM enters an infinite reset loop that can be exit only by power-off/power-on sequence. In failure is detected during self-tests, status is set to FAIL and error is returned.
HW integrity	HW integrity is guaranteed via check of HW sensors. If failure is detected during boot sequence, status is set to FAIL and error is returned.

4.1.2 Full self-tests list

Next table list of the tests performed in addition to the tests from Table 25: TPM integrity tests and Table 26: Cryptographic algorithm KATs on a TPM_SelfTestFull command execution.

Table 26: Cryptographic algorithm KATs

Algorithm tested	Test description
HMAC SHA1	HMAC-SHA1 computation on known data (16 bytes) / known key (16 bytes, same value as data) and comparison of output to the expected MAC (20 bytes)
KDF SP800-108	KDF (based on SHA1) computation on known data (16 bytes) / known label ("TEST") and comparison of output to the expected value (32 bytes).
Hash DRBG	Instantiate, Generate and Reseed API are tested in a single test sequence in accordance with §11.3 of **[SP800-90A]**. Output of HDRBG (55 bytes) is compared to a reference value.
AES	AES CFB encryption is done on known data (32 bytes) / known key (16 bytes) and known IV (16 bytes, same value as key). The 32 bytes output data are compared to the expected reference data. If comparison succeeds, AES CFB decryption is done on encrypted data with same key & same IV as encryption. 32 bytes output are compared to the initial plaintext data.
RSA	A known key is loaded (2048 bits length). Signature RSASSA-PKCS1-v1_5 is generated on known data (20 bytes). Output of signature is compared to a reference signature. If comparison is success, signature verification is performed. Failure state is entered if one of the step (generation or verification) fails.

4.1.3 *Conditional tests list*

Table 27: TPM conditional tests

Algorithm tested	Test description
Hash-DRBG	Each 32 bytes of generated data are compared to the previous generated data. If data are equal, status is set to FAIL and error is returned.
NDRNG	TPM performs AIS31 statistical test verification on NDRNG output. If test fails, status is set to FAIL and error is returned. Continuous self-tests are performed on the output of NDRNG (HW check).
FW load	During field upgrade procedure, several checks are performed before authorizing the FW to be upgraded: - Verification of signature (RSASSA-PSS) on the first data blob to ensure authentication of the FW - Verification of digest (SHA256) on each subsequent blob to guarantee integrity of the full FW.
RSA **key generation**	A new RSA key is generated or retrieved from pre-computed keys (done in BKG). Depending on the key purpose (signing or encrypting) indicated in TPM_KEY_USAGE structure, en/decryption or signing/verification is done on known data (16 bytes).

4.1.4 *Verification*

Successful completion of self-tests can be verified through use of TPM_GetTestResult command. If the first 4 bytes of response are equal to 0, self-tests completed successfully.

4.2 TPM2.0

Self-tests run by the cryptographic module are split in three categories:

- Power-up self-tests

- Full self-tests
- Conditional self-tests

The power-on self-tests do not require operator intervention in order to run. Power-on self-tests execution completes all tests except KATs on asymmetric algorithms (RSA, ECDSA, ECDH). Completion of power-on self-tests allows the TPM to be in a limited approved mode allowing to process commands that do not use asymmetric cryptography (see 1.7.2.1).

To switch from limited approved mode to full approved mode, operator shall execute TPM2_SelfTest(full = YES) command that will execute again the list of power-up self-tests plus the asymmetric cryptography self-tests listed in Table 29: Asymmetric cryptography self-tests list).

The security module outputs an "error" Return Code via the status interface when the error state is entered due to a failed self-test. While in error state, security module does not perform any cryptographic functions and all data output via the data output interface are inhibited.

If power-on self-tests have passed successfully, no status is indicated but commands that require self-tests to be completed can be successfully executed.

4.2.1 Power-up tests list

Table 28: Power-up self-tests list

Algorithm tested	Test description
SHA1	SHA1 computation on known data (16 bytes) and comparison of output to the expected digest (20 bytes)
SHA256	SHA256 computation on known data (16 bytes) and comparison of output to the expected digest (32 bytes)
HMAC SHA256	HMAC-SHA256 computation on known data (16 bytes) / known key (16 bytes, same value as data) and comparison of output to the expected MAC (32 bytes). Self-test allows validating the secure SHA algorithm also used in standalone (out of HMAC context).
KDF SP800-108	KDFa (based on SHA1) computation on known data (16 bytes) / known label ("TEST") and comparison of output to the expected value (32 bytes).
Hash-DRBG	Instantiate, Generate and Reseed API are tested in a single test sequence in accordance with §11.3 of **[SP800-90A]**. Output of HDRBG (55 bytes) is compared to a reference value.
AES	AES CFB encryption is done on known data (32 bytes) / known key (16 bytes) and known IV (16 bytes, same value as key). The 32 bytes output data are compared to the expected reference data. If comparison succeeds, AES CFB decryption is done on encrypted data with same key & same IV as encryption. 32 bytes output are compared to the initial plaintext data.
Triple-DES	Triple-DES CFB encryption is done on known data (32 bytes) / known key (24 bytes) and known IV (8 bytes). The 32 bytes output data are compared to the expected reference data. If comparison succeeds, Triple-DES CFB decryption is done on encrypted data with same key & same IV as encryption. 32 bytes output are compared to the initial plaintext data.
FW integrity	FW integrity is verified by computing an EDC (CRC-16 ISO 13239) and comparing it to reference values. FW integrity is verified during boot sequence before execution of one of the code block (CML, AFL and TPM) and during full self-tests execution. If failure is detected during boot sequence, TPM enters an infinite reset loop that can be exit only by power-off/power-on sequence. In failure is detected during self-tests, status is set to FAIL and error is returned.
HW integrity	HW integrity is guaranteed via check of HW sensors. If failure is detected during boot sequence, status is set to FAIL and error is returned.

4.2.2 *Full self-tests list*

Next table list of the tests performed in addition to the tests from Table 27: TPM conditional tests and Table 28: Power-up self-tests list on a TPM2_SelfTest(full=YES) command execution.

Table 29: Asymmetric cryptography self-tests list

Algorithm tested	Test description
RSA	A known key is loaded (2048 bits length). Signature RSASSA-PKCS1-v1_5 is generated on known data (20 bytes). Output of signature is compared to a reference signature. Signature verification is performed on the generated signature.
ECDH	A known private key d (32 bytes length) is used with a known point P of NIST P-256 curve to compute P = dQ. Q is compare to known reference point.
ECDSA	A known private key (256 bits) is used to generate ECDSA signature based on NIST P-256 curve. Output of signature is compared to a reference signature. Signature verification is performed on the generated signature.

4.2.3 *Conditional tests list*

Table 30: TPM conditional tests

Algorithm tested	Test description
FW integrity	FW integrity is verified by computing an EDC (CRC-16 ISO 13239) and comparing it to reference values.
Hash-DRBG	Each 32 bytes of generated data are compared to the previous generated data. If data are equal, status is set to FAIL and error is returned.
NDRNG	HW performs continuous tests on TRNG output. If test fails, bit TRNG_ERR is raised on SEC_STAT register and TPM enters failure mode.
FW load	During field upgrade procedure, several checks are performed before authorizing the FW to be upgraded: - Verification of signature (RSASSA-PSS) on the first data blob to ensure authentication of the FW - Verification of digest (SHA256) on each subsequent blob to guarantee integrity of the full FW.
RSA **key generation**	A new RSA key is generated or retrieved from pre-computed keys (done in BKG). Depending on the key purpose (signing or encrypting) indicated in sign attribute of the key, en/decryption or signing/verification is done on known data (16 bytes).
ECC key generation	On each ECC key generation, an ECDSA signature is generated and verified on curve NIST P-256.

4.2.4 *Verification*

Successful completion of self-tests can be verified through use of TPM2_GetTestResult command. The first 4 bytes of response indicate self-tests status. If they are equal to 0, self-tests completed successfully. If not, the subsequent 4 bytes indicate the list of algorithms not fully self-tested.

5 PHYSICAL SECURITY POLICY

The security module meets Physical Security protection requirements for FIPS level 1. CSPs are physically protected from unexpected disclosure and modification. Security module is tamper evident, encapsulated in an opaque package. Regular visual inspection must be conducted by user to check that HW integrity of the chip has not been damaged. Some physical security protection mechanisms beyond the requirements for level 1 have been implemented and are described in “Mitigations of other attacks”.

Normal operating ranges are defined in the respective TOE datasheet **[ST33TPHF2ESPI DS]**:

- **Temperature:**

  The normal operating temperature range of the security module is -40°C to +105°C.

- **Voltage:**

  The normal operating voltage range of the security module is 1.8V or 3.3V (±10%).

- **Frequency:**

  The internal system clock is created by an internal oscillator.

Operation outside these ranges is not guaranteed, but physical security mechanisms are implemented to assure that CSPs remain protected from unauthorized disclosure, usage, modification or deletion.

6 OPERATIONAL ENVIRONMENT

Module operational environment is “limited modifiable” because TPM FW can only be modified through field upgrade service (use of TPM_FieldUpgradeStart and TPM_FieldUpgradeData commands when TPM is executing in TPM1.2 mode and TPM2_FieldUpgradeStart and TPM2_FieldUpgradeData commands when TPM is executing in TPM2.0 mode). The Non-upgradable code blocks are non-modifiable.

FIPS 140-2 level 1 operational environment requirements of **[FIPS 140-2]** section 4.6.1 are then not applicable to the security module.

New firmware versions within the scope of this validation must be validated through the FIPS 140-2 CMVP. Any other firmware loaded into this module is out of the scope of this validation and require a separate FIPS 140-2 validation.

7 MITIGATIONS OF OTHER ATTACKS

The security module meets Physical Security protection requirements for FIPS level 1. Physical security at level 1 assumes no physical protection of CSPs. Physical security protection mechanisms beyond the level 1 requirements have been implemented and are described in this section.

7.1 Internal Tamper Detection

The security module contains an active metal shield that covers the internal TPM circuitry and memory components. Cutting, removing or modifying the shield layer will cause the TPM to Reset and enter a SHUTDOWN mode.

7.2 Environmental protection

The security module contains circuitry which will detect environmental conditions outside the range described in the product datasheet. Power supply voltage is continuously monitored. If conditions exist outside the range determined by the TPM tamper detection circuitry, the chip will reset and will enter a FAILURE mode. The chip will remain Reset and in FAIL mode as long as the environmental condition causing the tamper event persists.

8 References

Reference	Document
[ST33TPHF2ESPI DS]	ST33TPHF2ESPI Datasheet, STMicroelectronics, December 2015
[TPM1.2 Part1 r1.16]	TPM Main, Part 1, Design principles, Version 1.2 Level 2, rev 116, TCG
[TPM1.2 Part2 r1.16]	TPM Main, Part 2, TPM Structures, Version 1.2 Level 2, revision 116, TCG
[TPM1.2 Part3 r1.16]	TPM Main, Part 3, Commands, Version 1.2 Level 2, revision 116, TCG
[TIS 1.30]	TCG PC Client Specific TPM Interface Specification (TIS) – Version 1.3
[TPM2.0 Part1 r1.16]	TPM2.0 Main, Part 1, Architecture, rev 1.16, TCG
[TPM2.0 Part2 r1.16]	TPM2.0 Main, Part 2, Structures, rev 1.16, TCG
[TPM2.0 Part3 r1.16]	TPM2.0 Main, Part 3, Commands, rev 1.16, TCG
[TPM2.0 Part4 r1.16]	TPM2.0 Main, Part 4, Supporting routines, rev 1.16, TCG
[PTP 0.43]	TCG PC Client Platform TPM Profile (PTP) Specification, rev. 00.43 with errata
[FIPS 140-2]	FIPS PUB 140-2, Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST), CHANGE NOTICES (12-03-2002)
[FIPS DTR]	National Institute of Standards and Technology and Communications Security, *Derived Test Requirements(DTR) for FIPS PUB 140-2, Security Requirements for Cryptographic Modules*
[FIPS IG]	National Institute of Standards and Technology and Communications Security, *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*
[FIPS 180-4]	National Institute of Standards and Technology, *Secure Hash Standard*, Federal Information Processing Standards Publication 180-4, March 2012
[FIPS 186-4]	National Institute of Standards and Technology, *Digital Signature Standard (DSS),* Federal Information Processing Standards Publication 186-4, July 2013
[FIPS 197]	National Institute of Standards and Technology, *Advanced Encryption Standard (AES),* Federal Information Processing Standards Publication 197, November 2001
[SP800-135]	National Institute of Standards and Technology, *Existing Application-Specific Key Derivation Function Validation System*, September 2015.
[SP800-108]	National Institute of Standards and Technology, *Recommendation for Key Derivation Using Pseudorandom Functions*, October 2009.

Reference	Document
[SP800-131A]	National Institute of Standards and Technology, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, 11/06/15.
[FIPS 198-1]	National Institute of Standards and Technology, *The Keyed-Hash Message Authentication Code,* NIST Computer Security Division Page 3 07/26/2011, *(HMAC)*, Federal Information Processing Standards Publication 198-1, July, 2008
[SP800-90A]	National Institute of Standards and Technology, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, January 2012.
[SP800-38F]	National Institute of Standards and Technology, *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping,* December 2012.
[SP800-56A]	National Institute of Standards and Technology, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography,* March 2007.

9 ACRONYMS

Term	Definition
AES	Advanced Encryption Standard
CO	Crypto Officer
DES	Data Encryption Standard
DSAP	Delegate Specific Authorization Protocol
EK	Endorsement Key
FIPS	Federal Information Processing Standard
FUM	Field Upgrade Mode
GPIO	General Purpose I/O
HMAC	Keyed-Hashing for Message Authentication
NIST	National Institute of Standards and Technology
NV	Non-volatile (memory)
OIAP	Object-Independent Authorization Protocol
OSAP	Object Specific Authorization Protocol
PCR	Platform Configuration Register
RSA	Rivest Shamir Adelman
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
SHA	Secure Hash Algorithm
SPI	Serial Peripheral Interface
SRK	Storage Root Key
TCG	Trusted Computed Group
TPM	Trusted Platform Module
TSS	TPM Software Stack

IMPORTANT NOTICE – PLEASE READ CAREFULLY