Atmel Corporation

1150 E. Cheyenne Mountain Blvd.

Colorado Springs, Colorado 80906

Atmel Trusted Platform Module

AT97SC3204 / AT97SC3205

Firmware revisions: 1.2.29.01, 1.2.42.05, 1.2.42.06

Security Policy

FIPS 140-2, Level 1

Document Version 4.3

April 03, 2014

Revision History

Ver	Date	Author	Description
1.7	Oct 24, 2013	J. Hallman	FIPS 140-2 Certified AT97SC3204-X4 device
4.0	Mar 07, 2014	J. Hallman	Maintenance update: Addition of TPM revision AT97SC3205 to support SPI and I2C communication interfaces with the added support of Windows 8 Proof of Possession support. Add Heading numbers to all sections Globally replace AT97SC3204-X4 with AT97SC3204. Update supported package types and part numbers. Added physical Presence label to package drawing pins. Updated pin GPIO6 to GPIO-Express-00. Clarification of FIPS configurations. Updated Crypto Algorithm table and descriptions related to non-approved algorithms.
4.1	Mar 24, 2014	J. Hallman	1.1.1.1 Updates per evaluators comments.
4.2	Mar 27, 2014	J. Hallman	Corrected QFN part number error for TWI protocol
4.3	Apr 03, 2014	J. Hallman	Added redistribution note

Table of Contents

Tables:

Figures:

2 Module Overview

2.1 Definition of Cryptographic Module Security Policy

Security Level 1

The AT97SC3204 / AT97SC3205 Trusted Platform Module, referred to as the Target of Evaluation (TOE), 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 TOE implements version 1.2 of the Trusted Computing Group (TCG) specification for Trusted Platform Modules (TPM). The TPM is a single chip cryptographic 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, including the I/O pins through which all communication to and from the module occurs. The cryptographic boundary includes the module's internal 8-bit AVR microcontroller, 16KB of EEPROM, 128KB / 144KB of ROM, respectively, multiple banks of SRAM, cryptographic acceleration hardware and an internal Random Number Generator. Protection circuitry is included that can detect and respond to environmental changes and to hardware tamper events. No hardware or firmware components of the module are excluded from the requirements of FIPS 140-2.

The evaluated TOE has several configurations based upon the preferred external communication protocol. The LPC communication protocol is supported by devices AT97SC3204. The SPI and I2C communication protocol is supported by devices AT97SC3205. The communication protocol selection has no relevance to the security level of the device. All three protocol configurations are included in this evaluation and collectively referred to as the TOE.

Figure 1: AT97SC3204 Block Diagram (LPC Communication Protocol)

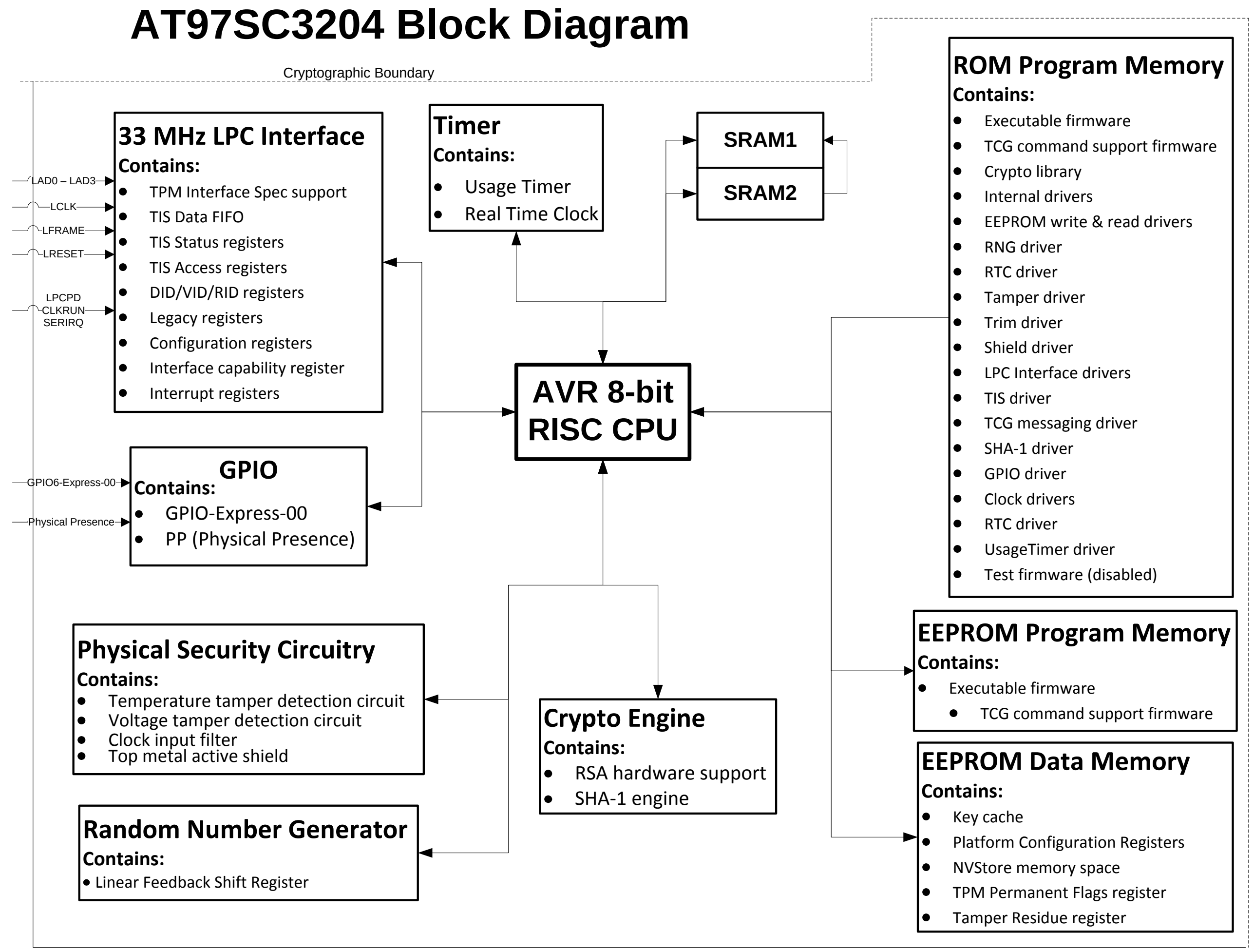
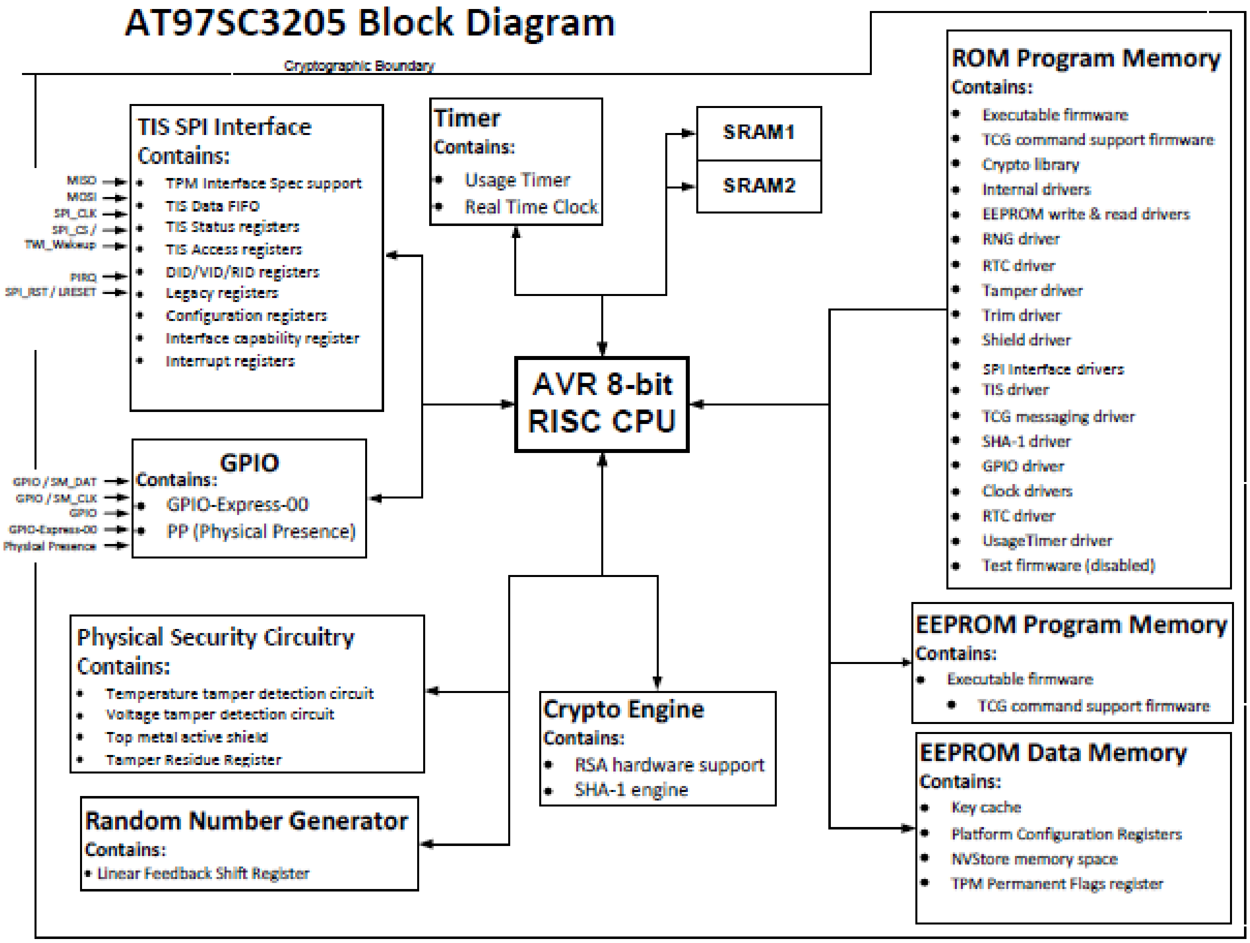


Figure 2: AT97SC3205 Block Diagram (SPI and I2C Communication Protocol)



The basic tasks of the TOE include the following:

- Measurement (through a SHA-1 hash mechanism), storage and reporting of the state of the computing platform bound physically and cryptographically to the platform.
- Execution of strong authentication mechanisms for identification of a computing platform identity.
- Provision of the following cryptographic services to the host platform:
  - Generation of RSA key pairs
  - RSA digital signature and verification
  - RSA key wrapping (encryption)
  - Random number generation

The TOE can be used by the host system to monitor the system boot process. The current system state may be compared to reference state values which were previously generated at a time when the system state was known to be trustworthy. The TPM can bind data or keys to a specific system status. The TPM can grant or deny access to keys and data if the current system state differs from the known trusted system state.

2.2 AT97SC3204 / AT97SC3205 TPM Implementation

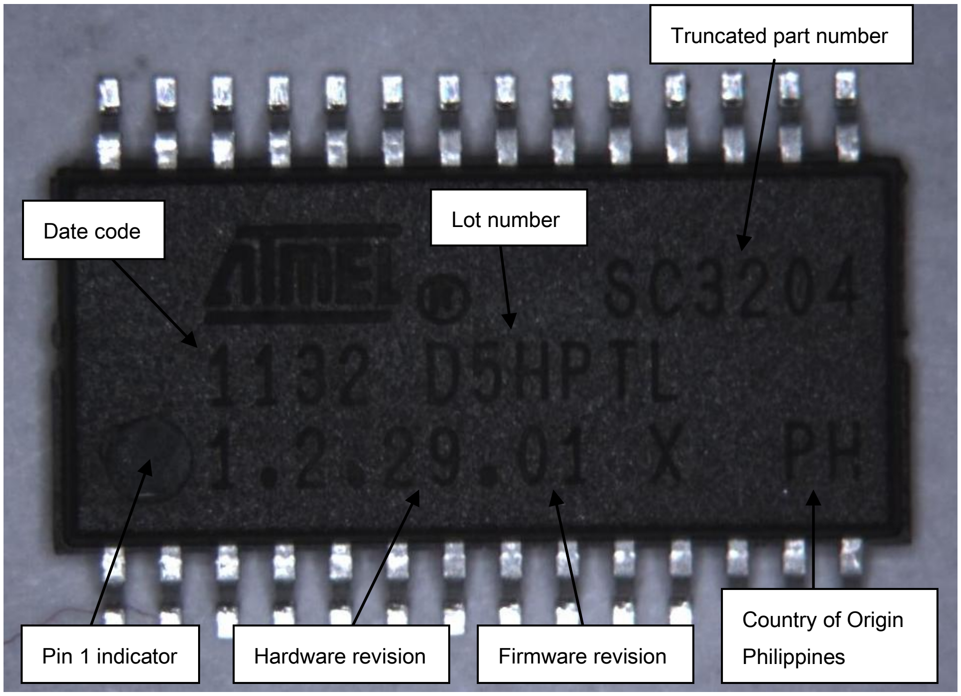
The Hardware and Firmware versions implemented in the FIPS evaluated and certified version of the TOE are:

- TOE Identification:
  - Trusted Platform Module Atmel AT97SC3204 / AT97SC3205
- TOE Version Identification:
  - AT97SC3204, LPC Communication Protocol: 1.2.29.01
  - AT97SC3205, SPI Communication Protocol: 1.2.42.05
  - AT97SC3205, I2C Communication Protocol: 1.2.42.06
- TOE Hardware Version Number:
  - AT97SC3204: 29
  - AT97SC3205: 42
- TOE Firmware Version Number:
  - AT97SC3204, LPC Communication Protocol: 01
  - AT97SC3205, SPI hardware interface activated: 05
  - AT97SC3205, I2C hardware interface activated: 06

The device name “AT97SC3204” is used throughout this security policy to refer to the TOE configured with the firmware image to support the LPC communication protocol unless specifically noted otherwise.

The device name “AT97SC3205” is used throughout this security policy to refer to the TOE configured with the corresponding firmware image to support the SPI or I2C communication protocol unless specifically noted otherwise.

Figure 3: Sample Image of the Physical Cryptographic Module

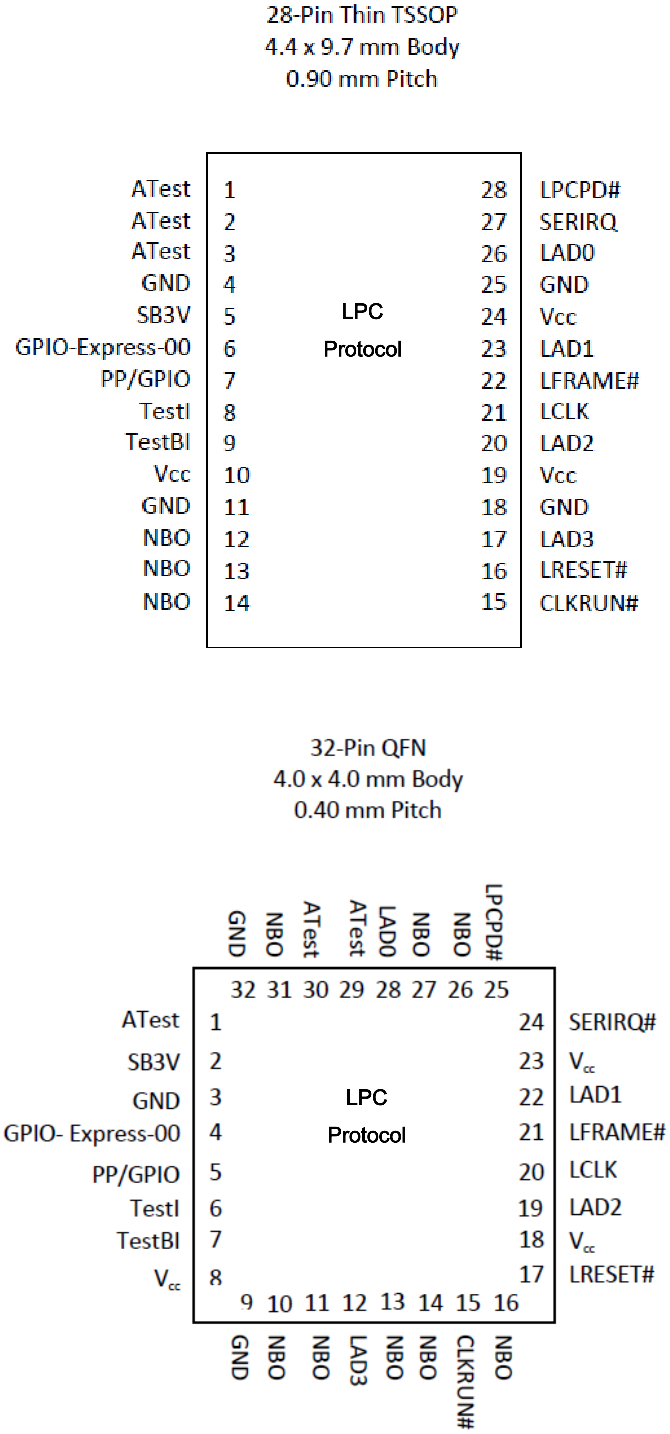


The TOE is manufactured in two packages:

- 28-pin 4.4mm TSSOP
    - LPC Communication Protocol
        - Part numbers: AT97SC3204-X4, AT97SC3204-U4
    - SPI Communication Protocol
        - Part numbers: AT97SC3205-X3, AT97SC3205-U3
    - I2C Communication Protocol
        - Part numbers: AT97SC3205T-X3, AT97SC3205T-U3
- 32-pin 4x4mm QFN
    - LPC Communication Protocol
        - Part numbers: AT97SC3204-G4, AT97SC3204-H4
    - SPI Communication Protocol
        - Part numbers: AT97SC3205-G3, AT97SC3205-H3
    - I2C Communication Protocol
        - Part numbers: AT97SC3205T-G3, AT97SC3205T-H3

The pin layout for the AT97SC3204 configuration of the TOE is shown in Figure 4.

Figure 4: AT97SC3204 Package Pinout Diagrams



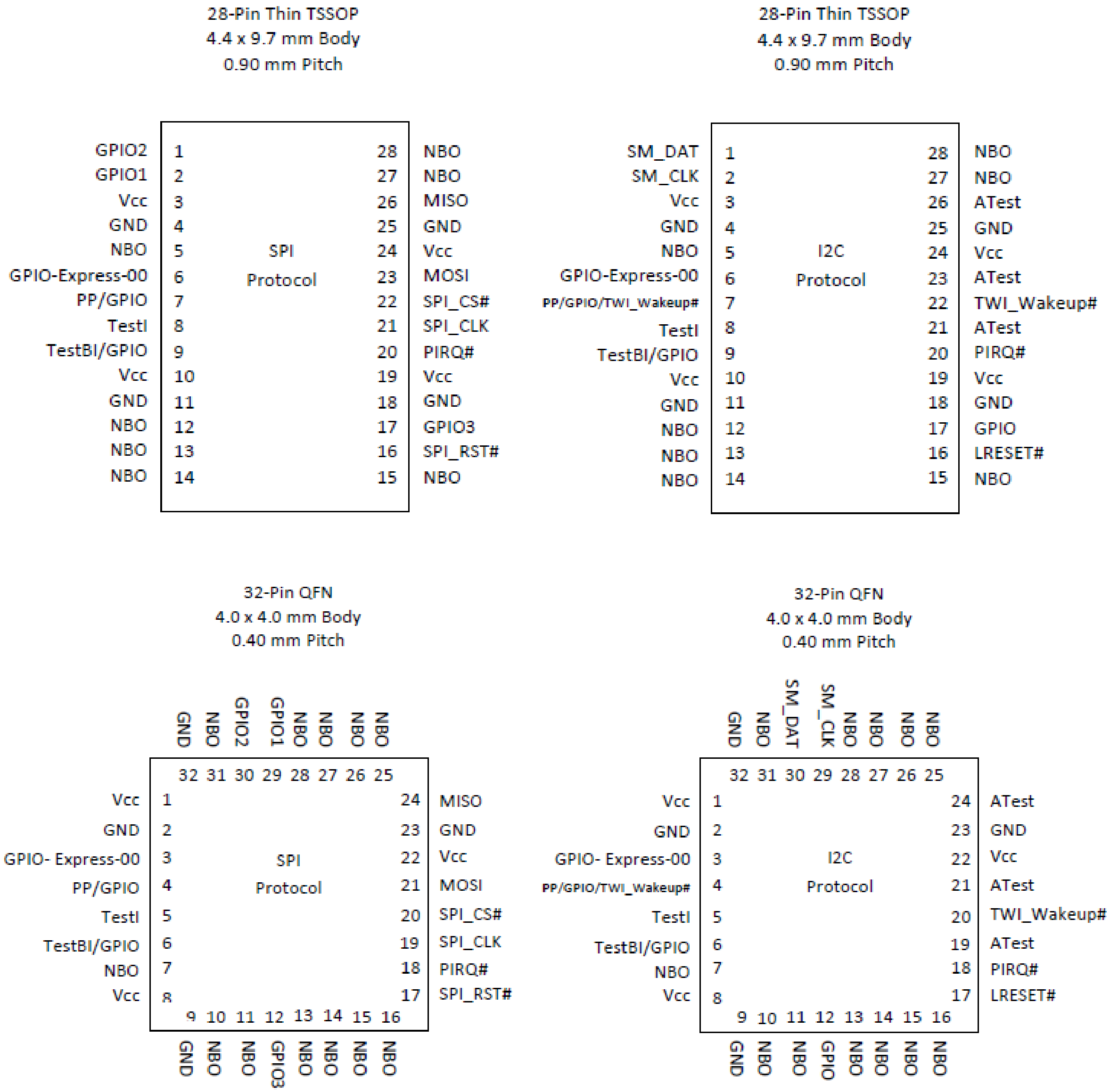
The pin description for the AT97SC3204 configuration of the TOE is given in Table 1.

Table 1 - AT97SC3204 Pin Definitions

TSSOP Pin	QFN Pin	LPC Pin Name	Pin type	Description
10,19,24	8,18,23	V_{CC}	I	3.3V Supply Voltage
5	2	SB3V	I	Standby 3.3V Supply Voltage
4,11,18,25	3,9,32	GND	I	Ground
16	17	LRESET#	I	PCI Reset Input Active Low
26	28	LAD0	I/O	LPC Command, Address, Data Line Input/Output
23	22	LAD1	I/O	LPC Command, Address, Data Line Input/Output
20	19	LAD2	I/O	LPC Command, Address, Data Line Input/Output
17	12	LAD3	I/O	LPC Command, Address, Data Line Input/Output
21	20	LCLK	I	33 MHz PCI Clock Input
22	21	LFRAME#	I	LPC FRAME Input
15	15	CLKRUN#	I	PCI Clock Run Input/Output
28	25	LPCPD#	I	LPC Power Down Input
27	24	SERIQ	I/O	Serialized Interrupt Request Input/Output
6	4	GPIO-Express-00	I/O	GPIO assigned to TPM_NV_INDEX_GPIO_00
8	6	TestI	I	Test Input (disabled)
9	7	TestBI	I	Test Input (disabled)
1,2,3	1,29,30	ATest	I	Atmel Test Pin (disabled)
7	5	PP/GPIO	I/O	GPIO assigned to Hardware Physical Presence
12,13,14	10,11,13,14 16,26,27,31	NBO	N/A	Not Bounded out – no connection to circuitry

The pin layout for the AT97SC3205 configuration of the TOE is shown in Figure 5.

Figure 5: AT97SC3205 Package Pinout Diagrams



The pin description for the AT97SC3205 configuration of the TOE is given in Table 2.

Table 2 - AT97SC3205 Pin Definitions

TSSOP Pin	QFN Pin	SPI Pin Name	I2C Pin Name	Pin type	Description
3,10,19,24	1,8,22	Vcc	Vcc	I	3.3V Supply Voltage
4,11,18,25	2,9,23,32	GND	GND	I	Ground
16	17	SPI_RST#	LRESET#	I	SPI: SPI Reset Pin I2C: Reset Input Active Low
21	19	SPI_CLK	ATest	I	SPI: SPI Clock Input I2C: Atmel Test Pin (disabled)
22	20	SPI_CS#	TWI_Wakeup#	I	SPI: SPI Chip Select I2C: Sleep State Pin
26	24	MISO	ATest	O	SPI: SPI Slave Data Output I2C: Atmel Test Pin (disabled)
21	21	MOSI	ATest	I	SPI: SPI Slave Data Input I2C: Atmel Test Pin (disabled)
2	29	GPIO1	SM_CLK	I/O	SPI: General Purpose Input / Output I2C: TWI Serial Clock Input
1	30	GPIO2	SM_DAT	I/O	SPI: General Purpose Input / Output I2C: TWI Serial Data Input / Output
17	12	GPIO3	GPIO	I/O	General Purpose Input / Output
20	18	PIRQ#	PIRQ#	I/O	PCI Interrupt Requests
6	3	GPIO-Express-00	GPIO-Express-00	I/O	GPIO assigned to TPM_NV_INDEX_GPIO_00
7	4	PP/GPIO	PP/GPIO/TWI_Wakeup#	I/O	GPIO assigned to Hardware Physical Presence
8	5	TestI	TestI	I	Test Input (disabled)
9	6	TestBI/GPIO	TestBI/GPIO	I	Test Input (disabled)
5,12,13,14, 15,27,28	7,10,11, 13-16, 25-28,31	NBO	NBO	N/A	Not Bounded out – no connection to circuitry

3 Security Level

Security Level 1

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**

4 Modes of Operation

Security Level 1

The Atmel TPM can operate in one of three permanently locked configurations; Legacy FIPS configuration, WIN8 FIPS configuration or Standard configuration. TPM configuration occurs one time during device initialization prior to taking ownership of the TPM device and is not reversible. Selection of a FIPS configuration results in this device being governed by this security policy. For a device configured as Standard configuration, this configuration is not the validated module and this security policy does not apply.

Critical firmware components are loaded and locked into the TPM internal EEPROM memory during the TPM manufacturing process. The device configuration is established during the final stage of the TPM system manufacturing process. The firmware revisions specified in the section entitled **AT97SC3204 / AT97SC3205 TPM Implementation** are the only firmware images that correspond to this security policy.

4.1 FIPS Operation Configurations

All firmware images of this security policy support a FIPS configuration. For ease of configuration identification, the configuration options are referred to as “Legacy FIPS” and “WIN8 FIPS”. The configurations only differ as defined below.

Legacy FIPS

- RSA keys of the key type TPM_KEY_LEGACY are not allowed. Requests to generate, load or use Legacy keys will result in failure of the command.
- All keys require authorization (cannot set key TPM_AUTH_DATA_USAGE value to TPM_AUTH_NEVER). RSA Key usage parameters will always require authorization for key generation, key loading or key usage. Requests to generate, load or use keys that do not require authorization will result in failure of the command
- TPM command TPM_Establish_Transport can only be executed with an algorithm ID of TPM_ALG_AES128 for encryption.

WIN8 FIPS

- RSA keys of the key type TPM_KEY_LEGACY are allowed. Requests to generate, load or use Legacy keys will **not** result in failure of the command. (For interoperability; all data processing with Legacy keys is considered equivalent to plaintext for the purpose of FIPS 140-2 validation.).
- The creation of keys requiring no authorization is allowed (can set key TPM_AUTH_DATA_USAGE value to TPM_AUTH_NEVER. Implicit authorization for users, allowed in FIPS 140-2 level 1).

- TPM command TPM_Establish_Transport can be executed with an algorithm ID of TPM_ALG_AES128 (for encryption) or TPM_ALG_MGF1 (For interoperability; this is considered equivalent to plaintext for FIPS 140-2 validation.).

The device configuration is established during the final stage of the TPM system manufacturing process and cannot be reversed. AT97SC3204 TPM units only support the "Legacy FIPS" configuration while the AT97SC3205 TPM units support both FIPS configurations.

4.2 TOE FIPS Indicators

The configuration of FIPS operation is recorded in the TPM_PERMANENT_FLAGS "FIPS" flag and the Atmel Specific TPM Command TPM_Lock_FIPS structure. Once set, the selected FIPS configuration cannot be disabled or altered for the lifetime of the TPM chip.

A User can verify the state of the "FIPS" flag by issuing the command TPM_getCapability with a capability name of TPM_CAP_FLAG (0x00000004) and a subcap of TPM_CAP_FLAG_PERMANENT (0x00000108).

A User can verify the internal FIPS configuration setting in the Atmel Specific TPM Command structure through the command TPM_Lock_FIPS with fipsMode = (0x00000002).

The below table identifies the permanent setting of the two control parameters used to lock the device in the selected configuration.

Table 4 - Configuration Identification Summary

Revision	Mode	Configuration	TPM_PERMANENT_ FLAGS "FIPS" Flag	TPM_Lock_FIPS Return Code Indication
AT97SC3204	FIPS	Legacy FIPS	TRUE	Legacy_FIPS
	Standard	Non-FIPS	FALSE	STD_MODE
AT97SC3205	FIPS	Legacy FIPS	TRUE	Legacy_FIPS
		WIN8 FIPS	FALSE	WIN_FIPS
	Standard	Non-FIPS	FALSE	STD_MODE

5 Ports and Interfaces

Security Level 1

The TOE provides the following physical ports and interface pins. For ease of deciphering pin functions between unit AT97SC3204 and unit SC97SC3205 versions, separate tables are provided for each.

Table 5 – AT97SC3204 Physical Ports and Interfaces

Pin Name	Pin type	VIL	VIH	VOL	VOH	Description
V_{CC}	I	-0.5V	3.63V			3.3V (+/- 10%) Power Supply Voltage
SB3V	I	-0.5V	3.63V			Standby 3.3V (+/- 10%) Supply Voltage
GND	I	0V	0V			System Ground
LRESET#	I	-0.5V	4.13V			Active Low PCI Reset Input. Driving this pin low resets the internal state of the TPM and is equivalent to removal of power from the chip.
LAD0, LAD1 LAD2, LAD3	I/O	-0.5V	4.13V	0.4V	2.5V	LPC Command, Address and Data Input/Output pins. The LPC bus transmits 8 bits per transaction in two 4-bit packets.
LCLK	I	-0.5V	4.13V			33 MHz PCI Clock Input. The frequency and duty cycle of this clock must be accurately maintained by the system to the parametric specifications listed in the T97SC3204 datasheet.
LFRAME#	I	-0.5V	4.13V			LPC FRAME indicator input pin
CLKRUN#	I/O	-0.5V	4.13V	0.4V	2.5V	PCI ClockRun Input/Output pin. Clock control handshake pin. When asserted by the system, indicates the intent to remove the 33MHz clock signal from the LCLK input pin. The TPM may hold off removal of the clock by holding the CLKRUN# pin low until it is prepared for removal of the clock signal.
LPCPD#	I	-0.5V	4.13V			LPC Power Down Input. LPCPD# is intended to indicate that the LPC Bus peripheral device (TPM) should prepare for system power-down, or for power to be shut off to devices on the LPC interface. Since the TPM automatically enters a low-power state after completion of every command, no special preparation is required by the TPM. No action is taken by the TPM when an active signal is received on this pin.
SERIQ	I/O	-0.5V	4.13V	0.4V	2.5V	Serialized Interrupt Request Input/Output. LPC Serialized IRQ. The Serial Interrupt Request pin is typically used to signal the system processor that the TPM has completed execution of a command and data is available to be read, when the TPM Locality has changed or when the TPM is ready to receive a new command. SERIRQ meets the Intel Low Pin Count (LPC) Interface Specification Revision 1.1 August 2002. This is an unused pin in general usage.
GPIO-EXPRESS-00	I/O	-0.5V	4.13V	0.4V	2.5V	General Purpose Input/Output pin configured by internal firmware as an Output only. Serves as an indicator signal by the TPM to other system components. The state of GPIO-EXPRESS-00 is determined by the data value written to the GPIO-EXPRESS-00 field by a TPM _NV_WriteValue or TPM_NV_WriteValueAuth command.
PP/GPIO	I	-0.5V	4.13V	0.4V	2.5V	General Purpose input/output pin whose state indicates the physical presence of an authorized operator.

Pin Name	Pin type	VIL	VIH	VOL	VOH	Description
TestI	I	-0.5V	4.13V			Test Input (permanently disabled during TPM manufacturing)
TestBI	I	-0.5V	4.13V			Test Input (permanently disabled during TPM manufacturing)
ATest	I	-0.5V	4.13V			Atmel Test Pin (permanently disabled during TPM manufacturing)

Table 6 – AT97SC3205 Physical Ports and Interfaces

Pin Name SPI / I2C	Pin type	VIL	VIH	VOL	VOH	Description
V_{CC}	I	-0.5V	3.63V			3.3V (+/- 10%) Power Supply Voltage
GND	I	0V	0V			System Ground
SPI_RST# / LRESET#	I	-0.5V	4.13V			Active Low PCI Reset Input. Driving this pin low resets the internal state of the TPM and is equivalent to removal of power from the chip.
SPI_CLK	I	-0.5V	4.13V			SPI interface clock. Asserted high for power savings when the TPM is not in use.
MISO	O			0.4V	2.5V	Master In Slave Out. SPI slave data output from the TPM.
MOSI	I	-0.5V	4.13V			Master Out Slave In. SPI slave data input to the TPM.
SPI_CS#	I	-0.5V	4.13V			SPI chip select input to the TPM.
SM_DAT	I/O	-0.5V	4.13V	0.4V	2.5V	I2C serial data input / output pin to the TPM.
SM_CLK	I	-0.5V	4.13V			I2C serial clock input to the TPM.
TWI_Wakeup#	I	-0.5V	4.13V			Sleep state pins. These two pins serves as the mechanism to wake the TPM from deep sleep.
PIRQ#	I/O	-0.5V	4.13V	0.4V	2.5V	PCI Interrupt Request line. Open drain output that pulls low to request an interrupt to the system processor.
GPIO-EXPRESS-00	I/O	-0.5V	4.13V	0.4V	2.5V	General Purpose Input/Output pin configured by internal firmware as an Output only. Serves as an indicator signal by the TPM to other system components. The state of GPIO-EXPRESS-00 is determined by the data value written to the GPIO-EXPRESS-00 field by a TPM _NV_WriteValue or TPM_NV_WriteValueAuth command.
PP/GPIO	I/O	-0.5V	4.13V	0.4V	2.5V	General Purpose input/output pin whose state indicates the physical presence of an authorized operator.
GPIO	I/O	-0.5V	4.13V	0.4V	2.5V	General Purpose input/output pin.
TestI	I	-0.5V	4.13V			Test Input (permanently disabled during TPM manufacturing)
TestBI	I	-0.5V	4.13V			Test Input (permanently disabled during TPM manufacturing)
ATest	I	-0.5V	4.13V			Atmel Test Pin (permanently disabled during TPM manufacturing)

6 Cryptographic Algorithms

The TOE supports the following cryptographic algorithms. Algorithm certificate numbers for each approved algorithm are listed.

Table 7 – Cryptographic Algorithms

7 Algorithm		Approved / Non-Approved	Certificate Number (AT97SC3204)	Certificate Number (AT97SC3205)
RSA	Digital Signature Verification, Digital Signature Generation when key length >=2048	Approved	**1203**	**1469**
	Digital Signature Generation when key length (n) is1024 <= |n| <2048	Non-Approved	N/A	N/A
SHA-1 (SHS)	Digital Signature Verification, Non-digital signature generation applications	Approved	**2015**	**2354**
	Digital Signature Generation	Non-Approved	N/A	N/A
HMAC		Approved	**1445**	**1757**
AES		Approved	**2333**	**2806**
RNG		Approved	**1163**	**1273**
TPM KDF (CVL)		Approved	Prior 2014	**250**
MGF1		Non-Approved	N/A	N/A

RSA: The TOE will permit RSA 1024-bit key generation and digital signature generation to retain device interoperability. These functions are Non-Approved and are considered equivalent to plaintext or obfuscation with no security claims.

The services that can employ this non-approved algorithm are:

- TPM_CreateWrapKey
- TPM_CMK_CreateKey
- TPM_CMK_ConvertMigration
- TPM_Sign
- TPM_CertifyKey
- TPM_CertifyKey2
- TPM_Quote
- TPM_Quote2
- TPM_TickStampBlob
- TPM_ReleaseTransportSigned
- TPM_ChangeAuthAsymStart

Note: These services also have the option to use approved 2048-bit key sizes.

SHA-1: The TOE will permit SHA-1 hash usage during digital signature generation to retain device interoperability. This function is Non-Approved and it is used with no security claims.

The services that employ this non-approved algorithm are:

  - TPM_Sign
  - TPM_CertifyKey
  - TPM_CertifyKey2
  - TPM_Quote
  - TPM_Quote2
  - TPM_TickStampBlob
  - TPM_ReleaseTransportSigned

MGF-1: The non-approved algorithm MGF-1 is not used to protect or encrypt CSPs. MGF-1 is used to obfuscate some input information passed to the module and some output information received from the module. The services that employ this non-approved algorithm are:

- Direct Anonymous Attestation commands
  - TPM_DAA_Join
  - TPM_DAA_Sign
- Transport commands
  - TPM_EstablishTransport
  - TPM_ExecuteTransport
  - TPM_ReleaseTransportSigned
- Seal commands
  - TPM_Sealx
  - TPM_unSeal
  - (note: MGF1 is not used by the service TPM_Seal)

8 Key Establishment

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

Table 8 – Security strength

Algorithm	Comparable number of bits of security
RSA-1024[1]	80
RSA-2048	112
AES-128	128

Note 1: RSA-1024 is for interoperability. This key size is non-compliant for FIPS 140-2 validation.

9 Security Rules

Security Level 1

The following sub-sections describe the security rules implemented by the TOE. The security rules are segregated into two categories:

- Security rules derived from the requirements of FIPS 140-2, the TCG TPM specification and PC Client TPM specification.
- Security rules imposed by Atmel

9.1 FIPS 140-2 Imposed Security Rules

1. The TOE supports the following interfaces (refer to Table 5 and Table 6):
    - Data input interface
        - Signals:
            - LPC: LAD0, LAD1, LAD2, LAD3
            - SPI: MOSI
            - I2C: SM_DAT
    - Data output interface
        - Signals:
            - LPC: LAD0, LAD1, LAD2, LAD3
            - SPI: MISO
            - I2C: SM_DAT
    - Control input interface
        - Signals:
            - LPC: PP/GPIO, CLKRUN#, LRESET#, LCLK, LFRAME#, SERIRQ, LPCPD#, LAD0, LAD1, LAD2, LAD3
            - SPI: PP/GPIO, SPI_RST#, SPI_CLK, SPI_CS#, PIRQ#, MOSI
            - I2C: PP/GPIO/TWI_Wakeup#, LRESET#, SM_CLK, SM_DAT, TWI_Wakeup#
    - Status output interface
        - Signals:

    - LPC: GPIO-Express-00, CLKRUN#, SERIRQ, LPCPD#, LAD0, LAD1, LAD2, LAD3
    - SPI: PIRQ#, MISO, GPIO-Express-00
    - I2C: PIRQ#, SM_DAT, GPIO-Express-00
  - Power input interface
    - Signals:
      - LPC: Vcc, SB3V, GND
      - SPI, I2C: Vcc, GND

2. The TOE interfaces are logically distinct from each other based on the command structure.
3. The logical state machine and command structure of the TOE inhibits all data output via the data output interface whenever an error state exists and while doing self tests.
4. Based on the structure of the commands, the TOE logically disconnects the output data path from the circuitry and processes performing the following key functions:
   - Key generation
   - Key zeroization
5. The TOE supports a Cryptographic Officer role and a User role.
6. When power is removed from the TOE, all existing authentication sessions are destroyed. Therefore, the TOE must re-authenticate every role or identity after each power-on sequence.
7. The TOE utilizes a production quality integrated circuit with standard passivation.
8. The TOE’s logical command structure, authentication mechanisms, memory management techniques, and physical implementation protect private and secret keys (Table 16 – CSP Identification) from unauthorized disclosure, modification and substitution.
9. The TOE’s logical command structure and authentication mechanisms protect public keys against unauthorized modification and substitution.

10. The TOE generates keys using a pseudo random number generator that is implemented in conformance to FIPS 186-2. National Institute of Standards and Technology, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-2, Appendix 3, Section 3.1 January 27, 2000.
11. The AT7SC3204 TPM's authentication mechanisms associate private and secret keys entered, stored, or output with the correct entity.
12. The TOE provides logical and/or physical mechanisms to zeroize all plaintext cryptographic keys and other unprotected critical security parameters within the TOE.
13. The TOE performs the following self tests:
    - Power-up and on-demand tests:
        - Cryptographic algorithm known-answer-tests (KAT).
            - SHA1
            - RSA
                a. Encrypt/decrypt
                b. Sign/Verify
            - RNG
            - HMAC
            - MGF1
            - AES (encryption and decryption)
        - Firmware integrity test.
    - Conditional tests:
        - Continuous random number generator tests.
            - The deterministic RNG produces blocks of 160 bits. Each subsequent 160-bit block output from the RNG is compared to the previous block. The test fails if any two compared 160-bit sequences are equal.
            - The nondeterministic RNG that provides entropy input to the deterministic RNG produces blocks of 160 bits.

Each subsequent 160-bit block output from the RNG is compared to the previous block. The test fails if any two compared 160-bit sequences are equal.

- Pair-wise consistency test for public and private keys

14. The power-on self-tests do not require operator intervention in order to run. Power-on self test execution always completes the full suite of self tests in its entirety. Input activity is ignored and output activity is inhibited until the self tests have successfully completed.
15. The TOE provides a “success” Return Code via the "status output" interface if all of the power-on self-tests have passed successfully.
16. The TOE outputs an “error” Return Code via the status interface when the error state is entered due to a failed self-test.
17. The TOE does NOT perform any cryptographic functions while in the error state.

9.2 Atmel Corporation Imposed Security Rules

1. The TOE provides a No Auth Required role that allows specific services to be performed without an open authentication session.
2. The TOE provides Role-based authentication.
3. The TOE supports concurrent operators and internally maintains the separation of the roles assumed by each operator and corresponding services using authentication sessions.
4. The TOE does NOT provide a Maintenance Role or Maintenance Interface.
5. Authentication mechanisms are required to support the following authorized roles:
   - *Crypto Officer*
   - *User*

The authentication mechanisms that support these roles meet the strength requirements of FIPS 140-2, Level 2.

- The authorized role ***Physical Presence*** does not require authentication mechanisms to support or enforce the role.
- The authorized role ***No Authentication Required*** does not require authentication mechanisms to support or enforce the role.

6. The TOE provides the following services:
   - Reference Table 10 – TOE Services
7. The TOE does NOT provide a bypass capability.
8. The TOE outputs plaintext key type data using the TMP_UnBind or TPM_UnSeal services.
9. The TOE's logical command structure, authentication mechanisms, memory management techniques, and physical implementation protect authentication data stored within the TOE against unauthorized disclosure, modification, and substitution.

10 Identification and Authentication Policy

Security Level 1

The following sub-sections describe the authentication and identification mechanisms employed by the TOE. The TOE provides Role-based authentication defined by FIPS 140-2. Note that authentication and identification mechanisms apply only to the CO and User roles.

10.1 Roles and Services

The following subsections describe the roles and services provided by the TOE.

10.1.1 TOE Roles

The TOE provides the following four authorized roles:

- **Crypto Officer (CO)** – The services provided under the CO Role require the operator to authenticate to the TOE as the “owner”. The CO services are used to initialize/configure the TPM and to install users.
- **User** – The services provided under the User Role require the operator to authenticate to the TOE as an “entity”. The User services obtain cryptographic or protected capability functions from the TPM.
- **Physical Presence (PP)** – A limited number of TPM commands may be authorized by assertion of Physical Presence (PP), which serves as an indication to the TPM that the operator is physically present at the system, not communicating over a network from a remote location. Authorization for assumption of the Physical Presence role is implicit. Physical Presence may be asserted in either of two methods, both of which authorize use of the same set of capabilities. Physical Presence authorization may be accomplished by setting a logic high voltage on the hardware pin at the time a service request is sent to the module (assertion of tpmGo bit in the TIS register [5]). The method for setting the voltage on the Physical Presence pin (e.g. connection to a button or a maintenance cover) is outside the cryptographic boundary. Physical Presence may also be asserted by sending the software command TSC_PhysicalPresence [4]. The capability for asserting software Physical Presence exists when the TPM is initially powered up. Atmel User Guidance instructs system designers to systematically disable the capability for asserting software Physical Presence after completion of essential early boot operations (e.g. at the completion of BIOS execution). Once disabled, Physical Presence may not be re-asserted by the software command mechanism until after a power down or Reset event. The capability to assert Physical Presence by the hardware pin mechanism remains active at all times and cannot be disabled.
- **No Authentication Required –** The authorized services provided under this role do not require any authentication. Authorization for assuming this role is implicit. The No Auth Required services do not require the use of protected capability functions (i.e. functions that require the use of CSPs associated with the CO or

User). The list of No Authentication Required services is included in the full list of TPM services in **Table 10 – TOE Services.** No Authentication Required services are identified by the label: **No Auth Required.**

Table 9 – TOE Authentication

Role	Type of Authentication	Authentication Data
Crypto Officer	Role based	160-bit auth data
User	Role based	160-bit auth data
Physical Presence	Implicit (none)	None
No Authentication Required	Implicit (none)	None

10.1.2 TOE Services

The services and corresponding roles provided by the TOE are shown in Table 10 for access control information.

Table 10 – TOE Services

	Services	Roles			
#	**Name**	**CO**	**User**	**PP**	**NoAuth**
1.	TPM_OIAP				X
2.	TPM_OSAP				X
3.	TPM_DSAP				X
4.	TPM_Terminate_Handle				X
5.	TPM_ChangeAuth		X		
6.	TPM_ChangeAuthOwner	X			
7.	TPM_TakeOwnership	X			
8.	TPM_Extend				X
9.	TPM_PcrRead				X
10.	TPM_PCR_Reset				X
11.	TPM_Quote		X		
12.	TPM_Quote2		X		

	Services	Roles			
#	**Name**	**CO**	**User**	**PP**	**NoAuth**
13.	TPM_DirWriteAuth	X			
14.	TPM_DirRead				X
15.	TPM_Seal		X		
16.	TPM_Sealx		X		
17.	TPM_Unseal		X		
18.	TPM_UnBind		X		
19.	TPM_CreateWrapKey		X		
20.	TPM_LoadKey		X		(X[1])
21.	TPM_LoadKey2		X		(X[1])
22.	TPM_EvictKey				X
23.	TPM_GetPubKey		X		
24.	TPM_OwnerReadPubek	X			
25.	TPM_OwnerReadInternalPub	X			
26.	TPM_CreateMigrationBlob		X		
27.	TPM_ConvertMigrationBlob		X		
28.	TPM_AuthorizeMigrationKey	X			
29.	TPM_MigrateKey		X		
30.	TPM_CMK_ApproveMA	X			
31.	TPM_CMK_CreateBlob		X		
32.	TPM_CMK_CreateKey		X		
33.	TPM_CMK_CreateTicket	X			
34.	TPM_CMK_SetRestrictions	X			
35.	TPM_CMK_ConvertMigration		X		

	Services	Roles			
#	Name	CO	User	PP	NoAuth
36.	TPM_SHA1Start				X
37.	TPM_ SHA1Update				X
38.	TPM_ SHA1Complete				X
39.	TPM_ SHA1CompleteExtend				X
40.	TPM_CertifyKey		X		
41.	TPM_CertifyKey2		X		
42.	TPM_Sign		X		
43.	TPM_GetRandom				X
44.	TPM_StirRandom				X
45.	TPM_SelfTestFull				X
46.	TPM_CertifySelfTest		X		
47.	TPM_ContinueSelfTest				X
48.	TPM_GetTestResult				X
49.	TPM_Reset				X
50.	TPM_SaveState				X
51.	TPM_StartUp				X
52.	TPM_OwnerClear	X			
53.	TPM_DisableOwnerClear	X			
54.	TPM_DisableForceClear				X
55.	TPM_GetCapability				X
56.	TPM_ GetCapabilityOwner	X			
57.	TPM_ DAA_JOIN	X			
58.	TPM_DAA_SIGN	X			

	Services	Roles			
#	Name	CO	User	PP	NoAuth
59.	TPM_SaveContext				X
60.	TPM_LoadContext				X
61.	TPM_NV_DefineSpace	X		X	
62.	TPM_NV_ReadValue	X		X	
63.	TPM_NV_ReadValueAuth		X	X	
64.	TPM_NV_WriteValue	X		X	
65.	TPM_NV_WriteValueAuth		X	X	
66.	TPM_OwnerSetDisable	X			
67.	TPM_PhysicalDisable			X	
68.	TPM_PhysicalEnable			X	
69.	TPM_PhysicalSetDeactivated			X	
70.	TPM_SetTempDeactivated		X	X	
71.	TPM_CreateEndorsementKeyPair				X
72.	TPM_CreateRevocableEK				X
73.	TPM_RevokeTrust			X	
74.	TPM_CreateCounter	X			
75.	TPM_ReadCounter				X
76.	TPM_ReleaseCounter		X		
77.	TPM_ReleaseCounterOwner	X			
78.	TPM_ReadPubek				X
79.	TPM_DisablePubekRead	X			
80.	TPM_OwnerReadPubek	X			
81.	TPM_MakeIdentity	X	X		

	Services	Roles			
#	Name	CO	User	PP	NoAuth
82.	TPM_ActivateIdentity	X	X		
83.	TPM_Delegate_CreateKeyDelegation		X		
84.	TPM_Delegate_CreateOwnerDelegation	X			
85.	TPM_Delegate_LoadOwnerDelegation	X			
86.	TPM_Delegate_Manage	X			
87.	TPM_Delegate_ReadTable				X
88.	TPM_Delegate_UpdateVerification	X			
89.	TPM_Delegate_VerifyDelegation				X
90.	TPM_EstablishTransport		X		
91.	TPM_ExecuteTransport		X		
92.	TPM_ReleaseTransportSigned		X		
93.	TPM_FlushSpecific				X
94.	TPM_ForceClear			X	
95.	TPM_GetTicks				X
96.	TPM_TickStampBlob		X		
97.	TPM_IncrementCounter		X		
98.	TPM_KeyControlOwner	X			
99.	TPM_ResetLockValue	X			
100.	TPM_SetCapability	X			
101.	TPM_SetOperatorAuth			X	
102.	TPM_SetOwnerInstall			X	
103.	TPM_SetOwnerPointer				X
104.	TPM_changeAuthAsymStart		X		

	Services	Roles			
#	Name	CO	User	PP	NoAuth
105.	TPM_changeAuthAsymFinish		X		
Locality Controlled Services					
106.	TPM_HASH_START				X
107.	TPM_HASH_DATA				X
108.	TPM_HASH_END				X
TPM Connection Services					
109.	TSC_PhysicalPresence				X
110.	TSC_ResetEstablishmentBit				X
Atmel Specific Services					
111.	TPM_SetState				X
112.	TPM_OwnerSetState	X			
113.	TPM_GetState				X
114.	TPM_Identify				X
115.	TPM_VerifySignature				X
116.	TPM_BindV20				X
117.	TPM_DeepSleep (I2C protocol only)				X
118.	TPM_Lock_FIPS				X

Note 1: For WIN8 FIPS configuration this command can be run with a NoAuth role.

Table 11 describes the TOE services.

Table 11 – TOE Service Description

#	Service Name	State Description
1.	TPM_OIAP	This service is used to create an OIAP authorization session. OIAP authorization sessions are generally used when use of multiple TPM entities are desired.
2.	TPM_OSAP	This service is used to create an OSAP authorization session. OSAP authorization sessions are generally used when use of a single TPM entity is required.
3.	TPM_DSAP	This User service creates an authorization session and a handle from a delegated AuthData value.
4.	TPM_Terminate_Handle	This User service is used to close an authorization session and clear the data associated with the session.
5.	TPM_ChangeAuth	This User service is used to modify the User's authorization key.
6.	TPM_ChangeAuthOwner	This CO service is used to modify the CO's authorization data stored on the TOE.
7.	TPM_TakeOwnership	This CO service is used to create the Storage Root Key (SRK) keypair and install the CO's identity/authorization data on the TOE.
8.	TPM_Extend	This service is used to update a Platform Configuration Register (PCR). A PCR consists of a 160-bit field that holds a cumulatively updated hash value and 4-byte status field.
9.	TPM_PcrRead	This service is used to read the contents of a specified PCR using non-cryptographic reporting.
10.	TPM_PCR_Reset	This User service is used to reset the contents of a specified PCR to default conditions.
11.	TPM_Quote	This User service is used to read the contents of a specified PCR using cryptographic reporting (digital signature).
12.	TPM_Quote2	This User service is used to read the contents of a specified PCR using cryptographic reporting (digital signature). Quote2 includes more detailed information about the platform configuration than TPM_Quote.
13.	TPM_DirWriteAuth	This CO service is used to provide write access to a specified Data Integrity Register (DIR).
14.	TPM_DirRead	This service is used to read the contents of a specified Data Integrity Register (DIR).
15.	TPM_Seal	This User service is used to input key type data and export it wrapped in a specified RSA public key. The Seal service outputs the wrapped key in an TOE specific method such that only the same TOE in the same state can perform a successful UnSeal service.
16.	TPM_Sealx	This User service is used to input encrypted key type data and export it wrapped in a specified RSA public key. The Seal service outputs the wrapped key in an TOE specific method such that only the same TOE in the same state can perform a successful UnSeal service.
17.	TPM_Unseal	This User service is used to decrypt and output key type data that was wrapped during a Seal service.

#	Service Name	State Description
18.	TPM_UnBind	This User service is used to unwrap and output key type data that was wrapped using a BindV20 service or using an external process that execute RSA public key encryption.
19.	TPM_CreateWrapKey	This User service is used to generate an RSA keypair, encrypt the private portion with a specified RSA public key, and output the public portion, key parameters and the wrapped private portion of the generated keypair.
20.	TPM_LoadKey	This User service is used to input, unwrap, and store an RSA keypair that resulted from the CreateWrapKey service, in order to make that key available for use by other services. Keys loaded into the TPM are identified by a handle that is assigned and output during execution of the TPM_LoadKey service. Subsequent operations will identify and locate this key by the reference handle. The TPM_LoadKey service includes the fixed handle in the key authorization Message Authentication data.
21.	TPM_LoadKey2	This User service is used to input, unwrap, and store an RSA keypair that resulted from the CreateWrapKey service, in order to make that key available for use by other services. Keys loaded into the TPM are identified by a handle that is assigned and output during execution of the TPM_LoadKey2 service. Subsequent operations will identify and locate this key by the reference handle. LoadKey2 does not include the handle in the key authorization Message Authentication data.
22.	TPM_EvictKey	This service is used to zeroize the contents of a specified key handle.
23.	TPM_GetPubKey	This User service is used to retrieve the public portion of a specified keypair.
24.	TPM_OwnerReadPubek	This CO service is used to export the public portion of the EK.
25.	TPM_OwnerReadInternalPub	This CO service is used to export the public portion of the EK or SRK.
26.	TPM_CreateMigrationBlob	This User service is used to migrate a private key from one TOE to another TOE. The migrated key is wrapped in a public key provided by the receiving TOE.
27.	TPM_ConvertMigrationBlob	This User service is used to store a migrated key resulting from the CreateMigrationBlob service.
28.	TPM_AuthorizeMigrationKey	This CO service is used to authorize a specific key for use in migration.
29.	TPM_MigrateKey	This User service serves as a migration authority. The service unwraps a key migration blob and re-wraps it with a public key provided to the TPM as a command parameter.
30.	TPM_CMK_ApproveMA	This CO service creates an authorization ticket that specifies an approved migration authority.
31.	TPM_CMK_CreateBlob	This User service is used to migrate a private key from one TOE to another TOE. The migrated key is wrapped in a public key provided by the receiving TOE. Similar to CreateMigrationBlob with more restrictions.

#	Service Name	State Description
32.	TPM_CMK_CreateKey	This User service both generates an asymmetric key and creates a secure storage bundle for that key. Migration of the new key is controlled by a migration authority.
33.	TPM_CMK_CreateTicket	This User service uses a public key to verify the signature over a digest, and generate a ticket to prove the verification.
34.	TPM_CMK_SetRestrictions	This CO service is used to restrict usage of a certified migration key with delegated authorization.
35.	TPM_CMK_ConvertMigration	This User service completes a migration sequence of a certified migration blob to a new TPM.
36.	TPM_SHA1Start	This service is used to start the process of calculating a SHA-1 hash.
37.	TPM_ SHA1Update	This service is used to continue the process of calculating a SHA-1 hash.
38.	TPM_ SHA1Complete	This service is used to finalize the process of calculating a SHA-1 hash.
39.	TPM_ SHA1CompleteExtend	This service is used to finalize the process of calculating a SHA-1 hash and extend the result into a specified PCR.
40.	TPM_CertifyKey	This User service is used to sign and output the public portion of a specified key.
41.	TPM_CertifyKey2	This User service is used to sign and output the public portion of a specified key when the certifying key requires usage authorization while the certified key does not.
42.	TPM_Sign	This User service is used to perform an RSA sign operation on the input data.
43.	TPM_GetRandom	This service is used to return a random number to the caller. The size of the random number is specified by the bytesRequested parameter.
44.	TPM_StirRandom	This service is used to add entropy to the state of the RNG.
45.	TPM_SelfTestFull	This service is used to initiate an operator requested Power-on Self Test (POST).
46.	TPM_CertifySelfTest	TPM_CertifySelfTest causes the TPM to perform a full self-test and return an authenticated value if the test passes.
47.	TPM_ContinueSelfTest	In the FIPS mode (set during the manufacturing process), this service just returns a ‘POST completed’ response code.
48.	TPM_GetTestResult	This service is used to retrieve POST test results. This service is also allowable in the Error state.
49.	TPM_Reset	This service is used to release all resources associated with all existing authorization sessions.
50.	TPM_SaveState	This service is used to save potentially volatile data into non-volatile memory.
51.	TPM_StartUp	This service is used to initiate a start-up of the TOE after a TPM_Init command.
52.	TPM_OwnerClear	This CO service is used to zeroize an TOE.
53.	TPM_DisableOwnerClear	This CO service is used to disable the OwnerClear service.

#	Service Name	State Description
54.	TPM_DisableForceClear	This service is used to disable the TPM_ForceClear service until the next start-up session. After TPM_DisableForceClear is executed, Owner keys and CSPs can be zeroized by executing TPM_ForceClear with Physical Presence authorization.
55.	TPM_GetCapability	This service is used to determine specific capabilities of an TOE based on the capArea and subCap parameters using non-cryptographic reporting.
56.	TPM_ GetCapabilityOwner	This CO service is used to retrieve all of the non-volatile and volatile flags in a single operation using non-cryptographic reporting.
57.	TPM_ DAA_JOIN	This CO service establishes the parameters for a Direct Anonymous Attestation procedure for a specific DAA issuing authority.
58.	TPM_DAA_SIGN	This CO service responds to a DAA challenge and proves the attestation held by a TPM without revealing the attestation held by that TPM.
59.	TPM_SaveContext	This service saves a set of context parameters from a loaded TPM outside the TPM, in order to reload the parameters at a later time.
60.	TPM_LoadContext	This service loads a previously saved context blob into a TPM.
61.	TPM_NV_DefineSpace	This CO service establishes space and access requirements for a nonvolatile storage space in a TPM.
62.	TPM_NV_ReadValue	This service is restricted to the CO if the NV area was set up to require Owner auth. If not, no auth is required. This service reads a value from a NV storage area and returns the value.
63.	TPM_NV_ReadValueAuth	This User service requires authorization, then reads a value from a NV storage area and returns the value.
64.	TPM_NV_WriteValue	This CO service writes a value to a predefined area in TPM nonvolatile memory.
65.	TPM_NV_WriteValueAuth	This CO service writes a value to a predefined area in TPM nonvolatile memory after completing authorization.
66.	TPM_OwnerSetDisable	This CO service is used to enable or disable an TOE by setting the value of the Disable Flag.
67.	TPM_PhysicalDisable	This service sets the permanent disable flag to TRUE. Requires assertion of Physical Presence.
68.	TPM_PhysicalEnable	This service sets the permanent disable flag to FALSE. Requires assertion of Physical Presence.
69.	TPM_PhysicalSetDeactivated	This CO service requires assertion of Physical Presence. The service changes the state of the persistent TPM deactivated flag.
70.	TPM_SetTempDeactivated	This service is used to make an TOE temporarily inactive without destroying the secrets protected by the TOE.
71.	TPM_CreateEndorsementKeyPair	This service is used to create a permanent internal Endorsement Key (EK) (an RSA public/private keypair) which is used in establishing the CO (owner) of the TOE.

#	Service Name	State Description
72.	TPM_CreateRevocableEK	This service is used to create a revokable internal Endorsement Key (EK) (an RSA public/private keypair) which is used in establishing the CO (owner) of the TOE.
73.	TPM_RevokeTrust	This service clears the Endorsement Key from the TPM, only if the EK were created using the service: TPM_CreateRevocableEK.
74.	TPM_CreateCounter	This service creates an internal TPM counter, establishes an initial counter value and establishes the access rules and AuthData for the counter.
75.	TPM_ReadCounter	This No Authentication Required service returns the current counter value.
76.	TPM_IncrementCounter	This service increments the indicated counter by one.
77.	TPM_ReleaseCounter	This User service releases a counter such that no Read or Increment commands will execute successfully.
78.	TPM_ReleaseCounterOwner	This CO service releases a counter such that no Read or Increment commands will execute successfully.
79.	TPM_ReadPubek	This service is used to export the public portion of the EK from the TOE.
80.	TPM_DisablePubekRead	This CO service is used to disable Users from exporting the public portion of the EK.
81.	TPM_OwnerReadPubek	This CO service is used to export the public portion of the EK.
82.	TPM_MakeIdentity	This CO service is used to create an identity within the TOE and output data necessary to complete attestation to that identity by a third party. A TPM identity is an alias to one and only one TPM Endorsement Key, which is cryptographically unique. A TPM identity key may sign status data generated internally by the TPM. The identity key may be certified by a third party process that takes place outside the TPM cryptographic boundary.
83.	TPM_ActivateIdentity	This CO service is used to activate an identity created within the TOE. Activation of a TPM identity occurs outside the TPM cryptographic boundary.
84.	TPM_Delegate_CreateKeyDelegation	This service delegates the privilege to use a key to another authorized User by creating a blob that can be used by TPM_DSAP.
85.	TPM_Delegate_CreateOwnerDelegation	This CO service delegates the CO's privilege to use a set of command ordinals, by creating a blob. Such blobs can be used as input data for TPM_DSAP or TPM_Delegate_LoadOwnerDelegation.
86.	TPM_Delegate_LoadOwnerDelegation	This CO service loads a delegate table row blob into an internal TPM non-volatile delegate table row.
87.	TPM_Delegate_Manage	This service establishes and/or manages the parameters and access privileges of a Delegation table.
88.	TPM_Delegate_ReadTable	This service loads a delegate table row blob into a non-volatile delegate table row. The service enables verification of delegation tables.

#	Service Name	State Description
89.	TPM_Delegate_UpdateVerification	This service sets the verificationCount in an entity (a blob or a delegation row) to the current family value, in order that the delegations represented by that entity will continue to be accepted by the TPM.
90.	TPM_Delegate_VerifyDelegation	This No Authentication Required service interprets a delegate blob and returns success or failure, depending on whether the blob is currently valid. The delegate blob is not loaded into the TPM.
91.	TPM_EstablishTransport	This User service establishes a Transport session. A TPM Transport session can provide a log of the commands within a session and can provide confidentiality of the commands within a session. Depending on the attributes specified for the session, this service may establish encryption keys and session logs. A shared secret may also be established that can be used to obfuscate either or both the input command data and the output Transport log. The session will be used by the service TPM_ExecuteTransport.
92.	TPM_ExecuteTransport	This service delivers a wrapped TPM command to the TPM where the TPM unwraps the command and then executes the command.
93.	TPM_ReleaseTransportSigned	This service completes a transport session. If logging for this session is turned on, then this command returns a digital signature of the hash of all operations performed during the session.
94.	TPM_FlushSpecific	TPM_FlushSpecific flushes from the TPM a specific handle (for a key, auth session, transport session or DAA session), and releases internal resources applied to that handle.
95.	TPM_ForceClear	This service is used to disable the ForceClear service until the next start-up session
96.	TPM_GetTicks	This service returns the current tick count of the TPM
97.	TPM_TickStampBlob	This service applies a time stamp to the Tick blob passed to the TPM. The TPM makes no representation regarding the blob, except that the blob was present at the TPM at the time indicated.
98.	TPM_KeyControlOwner	This CO service controls some attributes of keys that are loaded and stored within the TPM key cache, including enabling or disabling the ability to evict a loaded key.
99.	TPM_ResetLockValue	This CO service allows the TPM Owner exactly one opportunity to reset the TPM dictionary attack mitigation values while a timeout penalty is imposed.
100.	TPM_SetCapability	This CO service sets specific values in the TPM Capability register, which provide to outside entities various pieces of information regarding the design and current state of the TPM.
101.	TPM_SetOperatorAuth	This service allows the setting of the operator AuthData value. The operator AuthData value allows the execution of the TPM_SetTempDeactivated command.
102.	TPM_SetOwnerInstall	This Physical Presence service sets the PERMANENT flag that allows or disallows the capability to insert a TPM Owner.

#	Service Name	State Description
103.	TPM_SetOwnerPointer	This service will establish a reference to a specific secret the TPM will use when executing an Owner-secret-related OIAP or OSAP session. This command is only used to provide an Owner Delegation function for legacy code written for older TPM revisions which does not itself support Delegation.
104.	TPM_changeAuthAsymStart	This service starts the process of changing AuthData for an entity. It sets up an OIAP session that must be retained for use by the TPM_ChangeAuthAsymFinish command.
105.	TPM_changeAuthAsymFinish	This service completes the process that allows the owner of an entity to change the AuthData for the entity.
Locality Controlled Services		
106.	TPM_HASH_START	This No Authentication Required service begins a SHA-1 hash sequence. Execution of this service is limited to processes that are capable of calling TPM services that are restricted to a specific address range assigned to Locality 4. Atmel User guidance instructs system designers to restrict the Locality 4 address range to the pre-boot code that comprises the implicitly trusted Root of Trust for Measurement.
107.	TPM_HASH_DATA	This No Authentication Required service continues a SHA-1 hash sequence in Locality 4. This command (and optional subsequent HASH_DATA commands) contains the input data for the SHA-1 hash operation.
108.	TPM_HASH_END	This No Authentication Required service completes a SHA-1 hash sequence in Locality 4. This command terminates the input data for the SHA-1 hash operation and Extends the resulting digest into specific Platform Configuration Registers.
TPM Connection Services		
109.	TSC_PhysicalPresence	This service provides a capability to indicate a human's physical presence at the platform containing the TPM module. The ability to execute the TSC_PhysicalPresence command is restricted by Owner-controlled capabilities, which includes the ability to temporarily or permanently disable the service.
110.	TSC_ResetEstablishmentBit	This No Authentication Required service provides a nonvolatile indication that a TPM_HASH_START has been executed, which may indicate to the system that a trusted operating system has been loaded and verified.
Atmel Specific Services		
111.	TPM_SetState	This service is used to modify the state of the TPM State Identifier register. The State Identifier register controls and reports status of specific TPM capabilities like power loss status, tamper status and failed auth timeouts.
112.	TPM_OwnerSetState	This CO service is used to set specified internal latches of the TOE.

#	Service Name	State Description
113.	TPM_GetState	This service is used to retrieve state information from the TOE.
114.	TPM_Identify	This service is used to associate an TOE with an external host.
115.	TPM_VerifySignature	This service is used to verify a digital signature.
116.	TPM_BindV20	This RSA public key encryption service is used to wrap key type data with a public key provided to the TPM, and output the wrapped key blob. This is a public key operation and is therefore not a protected capability. TPM_BindV20 is provided as service for platforms or applications where RSA public key encryption capability does not exist outside the TPM cryptographic boundary.
117.	TPM_DeepSleep (I2C protocol only)	This service is used to allow the TPM to enter a low current state.
118.	TPM_Lock_FIPS	This service is used to query the device about its mode of operation

10.2 Authentication

The TOE supports role based authentication mechanisms for the roles defined above (see Roles and Services). Access to execute a TPM service that acts on a Critical Security Parameter requires that the operator assume a specific role. Assumption of the roles ***Crypto Officer*** and ***User*** is accomplished by completing one of the three TPM authentication processes described below. For the roles ***Physical Presence*** and ***No Auth Required*** authentication is implicit. Selection of the role takes place at the same time as authentication of the operator to the target TPM entity and assumption of the selected role.

Authorization data are held in protected storage within the TOE. The auth data are protected from unauthorized disclosure, modification or substitution.

The TOE invokes a two-step process for services requiring role/entity authentication. The first step is to open an authorization session. The second step is to authenticate to the entity that is to be used.

The TOE provides two protocols for opening an authorization session to authenticate to entities without revealing the entity authorization data over the

TOE's interface. A third protocol is supported which enables authentication by the TPM of a subject who has been delegated specific access privileges by the TPM Owner. The delegated subject may be given privileges to execute specific TPM Owner authorized commands. The TOE supports multiple sessions (concurrent operators) and uses these sessions plus individual service authentication to internally maintain the separation of the roles assumed by each operator and corresponding services. In all cases, the protocol exchanges nonce-data so that both sides of the transaction can compute a hash using shared secrets and nonce-data. Nonce-data are random numbers generated by the host and the TOE to prevent replay and man-in-the-middle attacks. The nonce-data values from the TOE are generated using the TOE's internal RNG. For convention, "odd" nonce-data values come from the Host and "even" nonce-data values come from the TPM (0 is an even number for this definition). The TOE enforces that the odd nonce-data value changes for each request. The TOE changes the value of the even nonce-data on each reply.

Each side generates the HMAC value and compares the internally generated HMAC value to the value received. The entity authorization data is protected from exposure on the communication bus using hashed objects sent over the interface.

The TPM authorization protocols are the following:

- Object-Independent Authorization Protocol (OIAP)
- Object-Specific Authorization Protocol (OSAP).
- Delegation-Specific Authorization Protocol (DSAP).

All authorization session protocols use a "rolling nonce-data" paradigm. This means that the TOE creates a new nonce-data value (i.e. random number) each time the TOE uses the session for an HMAC calculation. This "rolling nonce-data" paradigm ensures that the service requests are genuine and are not "replayed".

The first authorization session protocol is the OIAP, which enables the exchange of nonce-data with a specific TOE. Once an OIAP authorization session is established, its nonce-data can be included with the entity shared secret known by the operator as input to an HMAC computation. The authorization session can live indefinitely until either party requests the session termination. The OIAP protocol requires that re-authorization is completed before allowing execution of each requested service. The TPM_OIAP service starts the OIAP session.

The second protocol is the “Object Specific Authorization Protocol (OSAP)”. The OSAP allows establishment of an authentication session for a single entity. Once authorization has been completed successfully, multiple service requests for the authorized entity can be executed without re-execution of the authorization protocol. OSAP creates nonce-data that can authorize multiple commands without additional session-establishment overhead, but is bound to a specific entity by the secret auth value tied to the entity. The OSAP protocol verifies proof of knowledge of the entity secret through the HMAC protocol. The TPM_OSAP service starts the OSAP session. Once established, an OSAP session will remain open and valid until the either party terminates the session by resetting the parameter *continueAuthSession* in the service request command. The TPM_OSAP specifies the entity to which the authorization session is bound.

In general, the calculated authentication HMAC value is in the following forms:

- OIAP Session

  HMAC (Entity Authorization Data; SHA1 (inParams); inAuthSetupParams)

  where inParams = (ordinal, Input Arguments)

  where inAuthSetupParams = (Authorization Handle; authLastNonceEven; nonceOdd; continueAuthSession)

- OSAP Session

  HMAC (Shared Secret; SHA1 (inParams); inAuthSetupParams) where Shared Secret = HMAC (Entity Authorization Data; nonceEvenOSAP; nonceOddOSAP)

where inParams = (ordinal, Input Arguments)

where inAuthSetupParams = (authLastNonceEven; nonceOdd; continueAuthSession)

The HMAC value is a 20-byte number that includes multiple 20-byte random number components (2 random number components are used during the OIAP Session; 4 random number components are used during the OSAP Session). In this manner, the plaintext value of the actual Entity Authentication Data (20-byte number) is not revealed over the TOE interface.

The DSAP protocol allows the TPM Owner to create a new AuthData value and to delegate some selected TPM Owner privileges to that new AuthData value. As with OIAP and OSAP, the DSAP protocol requires an exchange of nonces to set up the authorization session. The TPM then uses the delegated AuthData value for all HMAC calculations. Once authorization of the delegated subject is completed by the TPM using the new AuthData value, usage of delegated privileges occurs using the same auth session protocols established for all authorized TPM commands.

10.2.1 Physical Presence

OIAP, OSAP and DSAP authorization protocols provide the authentication mechanism for assuming the roles of ***Crypto Officer*** or ***User***. A subset of the TPM capabilities may be executed after assuming the role of ***Physical Presence*** (PP). See Table 10 – TOE Services. While authorization is implicit for services requiring the Physical Presence role, the TPM requires assertion of the Physical Presence flag before allowing execution of these services. The capability for asserting the Physical Presence flag should exist only after the TPM is initially powered up or following a Reset event. Atmel User Guidance instructs system designers to systematically disable the capability for asserting the Physical Presence flag after completion of essential early boot operations. Once disabled, Physical Presence may not be re-asserted until after a power down or Reset event.

The list of TPM commands that can be authorized by assertion of Physical Presence:

- TPM_SetOwnerInstall
- TPM_PhysicalEnable
- TPM_PhysicalDisable
- TPM_PhysicalSetDeactivated
- TPM_SetTempDeactivated
- TPM_SetOperatorAuth
- TPM_ForceClear
- TPM_RevokeTrust
- TPM_NV_DefineSpace
- TPM_NV_WriteValue
- TPM_NV_WriteValueAuth
- TPM_NV_ReadValue
- TPM_NV_ReadValueAuth

10.2.2 Strength

The Entity Authentication Data is always a 20-byte number (160 bits), in which all bits are equally probable. Therefore, the "random attempt" probability of guessing the Authentication Data is 1 in 2^{160}. This probability applies to all TPM authorization protocols, OIAP, OSAP and DSAP.

10.2.2.1 AT97SC3204 TPM Strength

The AT97SC3204 communicates over a PCI bus [17] using an LPC protocol [16]. Each byte of command or data information transmitted to the TPM requires **15 LPC clock cycles**. Each LPC clock cycle requires a minimum of **29.5 nsec** for guaranteed operation within the Atmel AT97SC3204 datasheet limits. For the purposes of calculating data transmission rates for brute force authorization attempts at the highest frequency the internal circuitry is capable of supporting, a maximum frequency has been established by characterization of the product. While operation is not guaranteed at this frequency, some TPM chips are able to respond to data transmissions as high as 41 MHz, which corresponds to a minimum clock cycle time of **24.39 nsec**. This value can serve as the minimum possible cycle time for calculation of success probability during brute force repeated authorization attempts.

10.2.2.1.1 Strength calculation for OSAP

The shortest TPM command that requires authorization (TPM_OwnerClear) comprises **55 bytes** of command and data input, including the Owner authorization value. If the authorization protocol is OSAP and the Failed Authentication Attempts Counter has been disabled by the TPM Owner, repeated authentication attempts are allowed without re-initiating a new auth session. Receiving the return code from the TPM will always require a minimum of **10 bytes**. The minimum time required to input one attempted guess of a TPM OSAP authorization value is therefore:

15 cycles/byte * 24.39 nsec * (55 + 10 bytes) = 23.7802 μsec

The maximum number of attempts per minute is:

60 sec/23.7802 μsec = 2,523,102 maximum attempts per minute

The probability that multiple attempts to use the OSAP authentication mechanism during a one-minute period will succeed is:

2,523,102 / 2^{160} = 1.726376 x 10^{-42}

10.2.2.1.2 Strength calculation for OIAP

If the authorization session is an OIAP session, the TPM will require re-initiation of a new authorization session for every service request. This requires a minimum of **10 input bytes** followed by **38 bytes of output data**. To determine whether an authorization attempt has succeeded or failed, the OIAP session must be followed by execution of an authorized command. The shortest TPM command that requires authorization (TPM_OwnerClear) comprises **55 bytes** of command and data input, including the Owner authorization value. Receiving the return code from the TPM will always require a minimum of **10 bytes**. Execution of multiple failed authorization attempts will require that the TPM Failed Authentication Attempts Counter be disabled. The minimum time required to input one attempted guess of a TPM OIAP authorization value is therefore:

15 cycles/byte * 24.39 nsec * (55 + 10 + 10 + 38 bytes) = 41.341 µsec

The maximum number of attempts per minute is:

60 sec/41.341 µsec = 1,451,344

The probability that multiple attempts to use the OIAP authentication mechanism during a one-minute period will succeed is:

1,451,344 / 2^{160} = 9.930498 x 10^{-43}

10.2.2.1.3 Strength calculation for DSAP

The DSAP protocol behaves like OSAP, in that an authorization session targets a single TPM entity or service. However a DSAP authorization session will be automatically terminated at the conclusion of every failed attempt at authorization of a delegated service request. Setup of delegation tables and creation of delegation data blobs occurs before any attempted authorization of a delegated command, and therefore the time required for Delegation setup does not affect the probability that multiple authorization attempts executed during a one minute period would succeed. However, an authorization attempt executed using a DSAP can only be confirmed (pass or fail) by attempting execution of a delegated command. Failure of the delegated command terminates the DSAP session, so subsequent authorization attempts must re-execute a DSAP command to initiate a new session for each attempt.

The DSAP command requires a minimum of **145 input bytes**, including the delegation key blob or owner blob. The output of DSAP is **58 bytes.**

The shortest TPM command that requires authorization (TPM_OwnerClear) and can be delegated by the TPM owner comprises **55 bytes** of command

and data input, including the Owner authorization value. If the authorization protocol is DSAP and the delegation data blobs have been properly created (using either TPM_CreateKeyDelegation or TPM_CreateOwnerDelegation) then repeated DSAP authentication sessions are allowed without recreating the delegation data.

Receiving the return code from the TPM will always require a minimum of **10 bytes**. The minimum time required to input one attempted guess of a TPM OSAP authorization value is therefore:

15 cycles/byte * 24.39 nsec * (145 + 58 + 55 + 10 bytes) = 98.0478 µsec

The maximum number of attempts per minute is:

60 sec/98.0478 µsec = 611947

The probability that multiple attempt to use the OSAP authentication mechanism during a one-minute period will succeed is:

611,947 / 2^{160} = 4.187111 x 10^{-43}

Table 12 – AT97SC3204 Authentication Strength

Authentication Type	**Single Attempt Strength**	**Attempts per minute Strength** (calculations shown in text above)
OIAP	$1/2^{160}$ (6.842278×10^{-49})	9.930498×10^{-43}
OSAP	$1/2^{160}$ (6.842278×10^{-49})	1.726376×10^{-42}
DSAP	$1/2^{160}$ (6.842278×10^{-49})	4.187111×10^{-43}

10.2.2.2 AT97SC3205 TPM Strength

The AT97SC3205 communicates over a SPI bus using a TCG outlined SPI protocol [21] or a 2-wire interface “bit banged” over two GPIO ports. The 2-wire interface requires more clock cycles per byte of communication compared to the SPI interface, so only the SPI interface is considered in the below analysis.

At a minimum, to write one byte of command or data information transmitted to the TPM requires **40 SPI clock cycles**. Each SPI clock cycle requires a minimum of 41.6 nsec (24 MHz) and a maximum of 22.2 nsec (45 MHz) for guaranteed operation within the Atmel AT97SC3205 datasheet limits. For the purposes of calculating data transmission rates for brute force authorization attempts at the highest frequency the SPI bus MISO pad circuitry is capable of supporting, a maximum frequency has been established by characterization of the product. While operation is not guaranteed at this frequency, some TPM chips are able to respond to data transmissions at the higher **45 MHz** rate, which corresponds to a minimum clock cycle time of **22.2 nsec**. This value can serve as the minimum possible cycle time for calculation of success probability during brute force repeated authorization attempts.

10.2.2.2.1 Strength calculation for OSAP

The shortest TPM command that requires authorization (TPM_OwnerClear) comprises **55 bytes** of command and data input, including the Owner authorization value. If the authorization protocol is OSAP and the Failed Authentication Attempts Counter has been disabled by the TPM Owner, repeated authentication attempts are allowed without re-initiating a new auth session. Receiving the return code from the TPM will always require a minimum of **10 bytes**. The minimum time required to input one attempted guess of a TPM OSAP authorization value is therefore:

40 cycles/byte * 22.2 nsec * (55 + 10 bytes) = 57.720 µsec

The maximum number of attempts per minute is:

60 sec/23.7802 µsec = 1,039,501 maximum attempts per minute

The probability that multiple attempts to use the OSAP authentication mechanism during a one-minute period will succeed is:

1,039,501 / 2^{160} = 7.11255 x 10^{-43}

10.2.2.2.2 Strength calculation for OIAP

If the authorization session is an OIAP session, the TPM will require re-initiation of a new authorization session for every service request. This requires a minimum of **10 input bytes** followed by **38 bytes of output data**. To determine whether an authorization attempt has succeeded or failed, the OIAP session must be followed by execution of an authorized command. The shortest TPM command that requires authorization (TPM_OwnerClear) comprises **55 bytes** of command and data input, including the Owner authorization value. Receiving the return code from the TPM will always require a minimum of **10 bytes**. Execution of multiple failed authorization attempts will require that the TPM Failed Authentication Attempts Counter be disabled. The minimum time required to input one attempted guess of a TPM OIAP authorization value is therefore:

40 cycles/byte * 22.2 nsec * (55 + 10 + 10 + 38 bytes) = 100.344 µsec

The maximum number of attempts per minute is:

60 sec/100.344 µsec = 597,967

The probability that multiple attempts to use the OIAP authentication mechanism during a one-minute period will succeed is:

597,967/ 2^{160} = 4.09145 x 10^{-43}

10.2.2.2.3 Strength calculation for DSAP

The DSAP protocol behaves like OSAP, in that an authorization session targets a single TPM entity or service. However a DSAP authorization session will be automatically terminated at the conclusion of every failed attempt at authorization of a delegated service request. Setup of delegation tables and creation of delegation data blobs occurs before any attempted authorization of a delegated command, and therefore the time required for Delegation setup does not affect the probability that multiple authorization attempts executed during a one minute period would succeed. However, an authorization attempt executed using a DSAP can only be confirmed (pass or fail) by attempting execution of a delegated command. Failure of the delegated command terminates the DSAP session, so subsequent authorization attempts must re-execute a DSAP command to initiate a new session for each attempt.

The DSAP command requires a minimum of **145 input bytes**, including the delegation key blob or owner blob. The output of DSAP is **58 bytes.**

The shortest TPM command that requires authorization (TPM_OwnerClear) and can be delegated by the TPM owner comprises **55 bytes** of command and data input, including the Owner authorization value. If the authorization protocol is DSAP and the delegation data blobs have been properly created (using either TPM_CreateKeyDelegation or TPM_CreateOwnerDelegation) then repeated DSAP authentication sessions are allowed without recreating the delegation data.

Receiving the return code from the TPM will always require a minimum of **10 bytes**. The minimum time required to input one attempted guess of a TPM OSAP authorization value is therefore:

40 cycles/byte * 22.2 nsec * (145 + 58 + 55 + 10 bytes) = 237.984 µsec

The maximum number of attempts per minute is:

60 sec/237.984 µsec = 252117

The probability that multiple attempt to use the OSAP authentication mechanism during a one-minute period will succeed is:

$252117 / 2^{160} = 1.72505 \times 10^{-43}$

Table 13 – AT97SC3205 Authentication Strength

Authentication Type	**Single Attempt Strength**	**Attempts per minute Strength** (calculations shown in text above)
OIAP	$1/2^{160}$ (6.842278×10^{-49})	7.11255×10^{-43}

OSAP	$1/2^{160}$ (6.842278×10^{-49})	4.09145×10^{-43}
DSAP	$1/2^{160}$ (6.842278×10^{-49})	1.72505×10^{-43}

10.2.3 Failed Authentication Attempts Counter

The TOE accumulates the total number of failed authorization attempts on any entity in the FAILCOUNT register. A temporary lockout will occur if the number of attempts (A) is such that:

A mod <failure modulus> = 0.

The size of the <failure modulus> is set by specifying the FAILMOD parameter in the TPM_OwnerSetState service. The default value for FAILMOD is 3 (which allows 8 failed auth attempts before imposing a lockout penalty). FAILMOD may be modified using an Owner authorized command. Values greater than 10 are not permitted. Table 14 specifies the relationship between the FAILMOD parameter and the <failure modulus>.

Table 14 – Failure Modulus

FAILMOD	<failure modulus>
0	Disabled
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

The length of a lockout increases geometrically beginning with an initial lockout time equal to approximately 1.1 minutes. Table 14 shows the approximate lockout time given a specific FAILMOD and number of failed attempts.

Table 15 – Lockout delay vs. FAILMOD value

Lockout #	FAILMOD										Delay
	1	2	3	4	5	6	7	8	9	10	
1	2	4	8	16	32	64	128	256	512	1024	1.1 min
2	4	8	16	32	64	128	256	512	1024	2048	2.2 min
3	6	12	24	48	96	192	384	768	1536	3072	4.4 min
4	8	16	32	64	128	256	512	1024	2048	4096	8.8 min
5	10	20	40	80	160	320	640	1280	2560	5120	17.6 min
6	12	24	48	96	192	384	768	1536	3072	6144	35.2 min
7	14	28	56	112	224	448	896	1791	3584	7168	1.2 hr
8	16	32	64	128	256	512	1024	2048	4096	8192	2.3 hr
9 +	18	36	72	144	288	576	1152	2304	4608	9216	4.7 hr

10.2.3.1.1.1 If the TOE is in a locked-out condition, the TPM Owner is allowed exactly one attempt to reset the lockout, using the Owner-authorized command TPM_ResetLockValue. It is understood that this command allows the TPM owner to perform a dictionary attack on other authorization values by alternating a trial and this command. Similarly, delegating this command allows the owner's delegate to perform a dictionary attack.

When the chip is in a lockout condition, all commands other those in the following list will return the error code TPM_DEFEND_LOCK_RUNNING. If an unsuccessful TPM_OwnerSetState has occurred, then the TPM_OwnerSetState command will also return TPM_DEFEND_LOCK_RUNNING for the remainder of the lockout interval.

List of commands that may execute successfully while the TPM is in lockout condition:

TPM_ContinueSelfTest

TPM_FlushSpecific

TPM_Identify

TSC_PhysicalPresence

TSC_ResetEstablishmentBit

TPM_SHA1Complete

TPM_DirRead

TPM_GetCapability

TPM_OIAP

TPM_PCR_Reset

TPM_ResetLockValue

TPM_SHA1CompleteExtend

TPM_DSAP TPM_GetState

TPM_OSAP TPM_ReadCounter

TPM_SHA1Start

TPM_Startup

TPM_Extend

TPM_GetTestResult

TPM_OwnerSetState

TPM_Reset

TPM_SHA1Update

TPM_Terminate_Handle

10.3 Identification

To install TCG identities into the TOE, the CO performs the TPM_MakeIdentity and TPM_ActivateIdentity services. The concept of a TCG Identity in a TPM does not correspond to the definition of Identity in FIPS 140-2. A TPM Identity is an alias to one and only one TPM Endorsement Key, which is cryptographically unique. Creation of a TPM identity (TPM_MakeIdentity) invokes creation of a corresponding Identity key. A TPM Identity key may sign status data generated internally by the TPM. The identity key may be certified by a third party process that takes place outside the TPM cryptographic boundary.

11 Access Control Policy

Security Level 1

The following sub-sections describe the identification and usage of Critical Security Parameters (CSPs).

11.1 Definition of Critical Security Parameters (CSPs)

Table 16 – CSP Identification describes the CSPs that may reside in the module. Services that create, modify or make use of CSPs are listed in the Services column, which references services listed in Table 10 – TOE Services.

Table 16 – CSP Identification

CSP	Algorithm	Size (bits)	Description	Services (listed in **Table 10**)
Endorsement Key (EK)	RSA	2048	Internally generated 2048-bit RSA key pair. Used to decrypt Owner auth value and Identity certificates. Can be zeroized only if it is generated using the service TPM_createRevokableEK.	7, 24, 71, 72, 73, 78, 79, 80, 82,
Storage root Key (SRK)	RSA	2048	Key-encrypting key. Root key for the TPM storage hierarchy. Internally generated by the service TPM_takeOwnership. Child keys of the SRK may be created in the TPM Storage hierarchy using the service createWrapKey.	6, 7, 25, 81
Private Storage Keys	RSA	2048	RSA private keys used in unwrapping operations. Usage properties are determined by the Key property values stored and protected with the key data. Key data determines key usage properties that may restrict key usage for Encryption or Signature services. Key Migration capability.	7, 15, 16, 17, 18, 19, 20, 21, 22, 29, 31, 32, 35, 82, 88, 89, 91
Seal Keys	RSA	2048	RSA private Storage Keys used to unwrap Sealed data blobs. Unwrapping operations require verification of TPM state in addition to standard user authentication.	15, 16, 19, 20, 21
AuthChange Keys	RSA	1024 2048	An ephemeral key whose use is restricted to providing confidentiality for new authentication data in the process of changing authentication values.	104, 105
Private Signature Keys. Private Identity Keys	RSA	1024 2048	RSA private keys used in digital signature operations. Usage properties are determined by the Key property values stored and protected with the key data.	11, 12, 15, 16, 19, 22, 40, 41, 42, 46, 81, 82, 92
Private Encryption Keys. Bind	RSA	1024 2048	RSA private keys used to encrypt key type data for storage outside the cryptographic boundary. Created by the service	15, 16, 17, 18, 19, 20, 21

Keys			TPM_createWrapKey. Not allowed to perform Signature operations.	
Migrate Keys	RSA	1024 2048	TPM_MigrateKey decrypts an input packet (coming from TPM_CreateMigrationBlob or TPM_CMK_CreateBlob) and then re-encrypts it with a public key that was input with the command.	26, 27, 28, 29, 30, 31, 32, 33, 34, 35
tpmProof	N/A	160	Internally generated secret value used to verify proof of origin for encrypted objects created by the TPM and stored outside the module.	3, 7, 15, 17, 19, 20, 21, 16, 26, 28, 30, 32, 33, 31, 35, 41, 51, 58, 59, 80, 82, 83, 84, 87, 88
contextKey	AES	128	Symmetric key used to provide internal encryption and decryption of Context blobs.	59, 60
delegateKey	AES	128	Symmetric key used to encrypt and decrypt sensitive data passed to the module with Delegation service requests.	83, 84, 85, 86, 87, 88, 89
daaBlobKey	AES	128	Symmetric key used to encrypt sensitive data used to build a TPM_DAA_TPM structure that is exported from the module	57, 58
transportKey	AES	128	Symmetric key used to encrypt parameters from TPM commands that are passed to the module within a Transport session	90, 91, 92
Activate Identity Key	AES	128	Symmetric session key created by a Certification authority, passed to the module as an encrypted parameter of TPM_activateIdentity and used to decrypt the TPM_IDENTITY_CREDENTIAL.	81
HMAC Keys	HMAC	160	SHA-1 HMAC keys used in authorization protocols. The HMAC key is specified by the TPM_AUTHDATA description in an OIAP or Transport session. For an OSAP or DSAP session, the HMAC key is the shared secret that was calculated during the session setup.	5, 6, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 40, 41, 42, 46, 52, 53, 56, 57, 58, 61, 62, 63, 64, 65, 66, 77, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 91, 92, 96, 97, 98, 99, 100, 110
Platform Configuration Registers (PCRs)	SHA-1	160	An array of 160-bit registers held in EEPROM or RAM memory as specified in the TCG TPM PC Client TPM specification [5]. A PCR will contain SHA-1 hash results from measurements of the internal TPM state and the external system state.	7, 8, 10, 11, 12, 15, 16, 17, 19, 21, 31, 32, 39, 40, 41, 51, 62, 63, 64, 81, 82, 87,
Authentication data	HMAC	160	Auth data held in protected EEPROM storage for permanent TPM entities (Owner and SRK) and for TPM entities that are not cryptographic keys, such as counters, NVStorage indices, Transport sessions, Delegations sessions.	5, 6, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 40, 41, 42, 46, 52, 53,

				56, 57, 58, 61, 62, 63, 64, 65, 66, 77, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 91, 92, 96, 97, 98, 99, 100, 110
Monotonic Counters	N/A	N/A	Continuously increasing counter. Always readable, but can't be modified, reset or deleted by any external or internal process.	52, 74, 76, 77, 97
Tick Counter	N/A	N/A	Internal counter, reset on power-up. Records the number of clocks since the start of a Tick session.	90, 91, 92, 95, 96,
NonVolatile Storage	N/A	N/A	Protected nonvolatile EEPROM memory reserved for user data.	61, 64, 65,
RNG Seed	SHA-1	160	Input to RNG. Replaced by the new RNG output value each time the RNG is called.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115
RNG Seed Key	SHA-1	160	Input to RNG. Receives input from nondeterministic entropy generator.	
Permanent Data, state flags, tamper registers and critical data	N/A	N/A	Data, symmetric keys, secrets and state flags held in TPM protected EEPROM memory. These include: revMajor, revMinor, ekReset, operatorAuth, authDIR, SRKauth, ownerAuth, contextKey, pcrAttrib, delegateKey, tpmProof, FIPS indicator, rngState, familyTable, delegateTable, lastFamilyID, noOwnerNVWrite, restrictDelegate, tpmDAAseed, daaProof, daaBlobKey, deactivated, disableForceClear, PhysicalPresence, physicalPresenceLock, bGlobalLock. The Tamper registers store a record of tamper events recorded if operation outside the specified environmental conditions is detected (voltage, temperature, frequency, hardware shield).	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,

				90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115
KAT values	N/A	N/A	Protected stored values used by power-up Known Answer Tests. Includes fixed results for RNG, RSA, AES, HMAC, SHA-1, ROM integrity test, EEPROM integrity test.	45, 46
Executable firmware	N/A	N/A	Executable software stored in protected TPM memory (both ROM and EEPROM). Contains support for execution of all TPM services including cryptographic operations.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115

11.2 Definition of Public Keys

The following Public Keys may be contained within the module

Public Key	Description
Signature verification Key (public portion of Endorsement Key, Signing Key, Identity Key)	RSA public/private key pairs may be loaded into the TPM and made available for use by other TPM capabilities. Usage is gated by TPM access control mechanisms specified by parameters included with each loaded key pair. Reading the Public portion of a loaded key may or may not require authorization of the key Owner. Loaded public keys may be used to verify digital signatures. The public Endorsement Key is held within TPM protected memory space.
RSA encryption key (public portion of Storage Key, Seal Key, AuthChange Key, Migrate Key)	Loaded RSA public Storage keys may be used as Parent keys to encrypt data blobs to be exported for storage of Child keys and key data outside the cryptographic boundary. Loaded and authorized public Storage keys may be used to create encrypted Migration blobs that are exported from the TPM and may be subsequently imported by a different TPM containing the corresponding private key. A loaded public key may be used to certify that another key residing in the TPM is protected by the TPM and will never be revealed.
BindV20 public key	This RSA public key is used to wrap arbitrary unprotected data with a public key provided to the TPM, and output the wrapped data blob. This is a public key operation and is therefore not a protected capability. The BindV20 public key is entered into the TPM as a plaintext parameter of the TPM_BindV20 service. The private key associated public key is not loaded into the TPM at the time the public key encryption operation takes place. TPM_BindV20 is provided as an unprotected service for platforms or applications where RSA public key encryption capability does not exist outside the TPM cryptographic boundary.

11.3 CSP Access Type

Table 17 describes the CSP access types.

Table 17 – CSP Access Type

Access Type	Description
Generate Private (G_p)	“Generate Private” is defined as the creation of an RSA key pair.
Generate Secret (G_s)	“Generate Secret” is defined as the creation of a Secret.
Sign (S)	“Sign” is defined as the process in which an RSA private key is employed to generate a digital signature.
Key Unwrap (K_u)	“Key Unwrap” is defined as the process in which an RSA private key is employed to decrypt a Secret.
Key Wrap (K_w)	“Key Wrap” is defined as the process in which an RSA public key is employed to encrypt a Secret or RSA private key.
Use (U)	“Use” is defined as a process that uses a Secret.
Delete (D)	“Delete” is defined as the zeroization of an RSA private key.

11.4 Logical Access Policy

Table 18 describes the Logical Access policy.

Table 18 – Logical Access

Services		Roles				CSPs	
#	Name	CO	User	PP	NoAuth	Private	Secret
1.	TPM_OIAP				X	-	-
2.	TPM_OSAP				X	-	-
3.	TPM_DSAP				X	-	-
4.	TPM_Terminate_Handle				X	-	-
5.	TPM_ChangeAuth		X			-	U ; G_s
6.	TPM_ChangeAuthOwner	X				-	U ; G_s
7.	TPM_TakeOwnership	X				K_u ; G_p	U ; G_s
8.	TPM_Extend				X	-	-
9.	TPM_PcrRead				X	-	-
10.	TPM_PCR_Reset				X	-	-
11.	TPM_Quote		X			S	U
12.	TPM_Quote2		X			S	U
13.	TPM_DirWriteAuth	X				-	U
14.	TPM_DirRead				X	-	-
15.	TPM_Seal		X			S	U ; K_w
16.	TPM_Sealx		X			S	U;$K_{w;}K_u$
17.	TPM_Unseal		X			K_u	U
18.	TPM_UnBind		X			K_u	U
19.	TPM_CreateWrapKey		X			G_p	U ; K_w
20.	TPM_LoadKey		X		(X[1])	K_u	U
21.	TPM_LoadKey2		X		(X[1])	K_u	U
22.	TPM_EvictKey				X	D	-
23.	TPM_GetPubKey		X			-	U
24.	TPM_OwnerReadPubek	X				-	U
25.	TPM_OwnerReadInternalPub	X				-	U
26.	TPM_CreateMigrationBlob		X			K_w	U
27.	TPM_ConvertMigrationBlob		X			K_w	U
28.	TPM_AuthorizeMigrationKey	X				-	U
29.	TPM_MigrateKey		X			K_u	U; K_w
30.	TPM_CMK_ApproveMA	X				-	U
31.	TPM_CMK_CreateBlob		X			K_w	U
32.	TPM_CMK_CreateKey		X			G_p	U; K_w
33.	TPM_CMK_CreateTicket	X				-	U
34.	TPM_CMK_SetRestrictions	X				-	U

Services		Roles				CSPs	
#	Name	CO	User	PP	NoAuth	Private	Secret
35.	TPM_CMK_ConvertMigration		X			-	U; K_w
36.	TPM_SHA1Start				X	-	-
37.	TPM_ SHA1Update				X	-	-
38.	TPM_ SHA1Complete				X	-	-
39.	TPM_ SHA1CompleteExtend				X	-	-
40.	TPM_CertifyKey		X			S	U
41.	TPM_CertifyKey2		X			S	U
42.	TPM_Sign		X			S	U
43.	TPM_GetRandom				X	-	-
44.	TPM_StirRandom				X	-	-
45.	TPM_SelfTestFull				X	-	-
46.	TPM_CertifySelfTest		X			S	U
47.	TPM_ContinueSelfTest				X	-	-
48.	TPM_GetTestResult				X	-	-
49.	TPM_Reset				X	-	-
50.	TPM_SaveState				X	-	-
51.	TPM_StartUp				X	-	-
52.	TPM_OwnerClear	X				D	U
53.	TPM_DisableOwnerClear	X				-	U
54.	TPM_DisableForceClear				X	-	-
55.	TPM_GetCapability				X	-	-
56.	TPM_ GetCapabilityOwner	X				-	U
57.	TPM_ DAA_JOIN	X				-	U
58.	TPM_DAA_SIGN	X				-	U
59.	TPM_SaveContext				X	-	-
60.	TPM_LoadContext				X	-	-
61.	TPM_NV_DefineSpace	X		X		-	U
62.	TPM_NV_ReadValue	X		X		-	U
63.	TPM_NV_ReadValueAuth		X	X		-	U
64.	TPM_NV_WriteValue	X		X		-	U
65.	TPM_NV_WriteValueAuth		X	X		-	U
66.	TPM_OwnerSetDisable	X				-	U
67.	TPM_PhysicalDisable			X		-	-
68.	TPM_PhysicalEnable			X		-	-
69.	TPM_PhysicalSetDeactivated			X		-	-
70.	TPM_SetTempDeactivated		X	X		-	U
71.	TPM_CreateEndorsementKeyPair				X	G_p	-

Services		Roles				CSPs	
#	**Name**	**CO**	**User**	**PP**	**NoAuth**	**Private**	**Secret**
72.	TPM_CreateRevocableEK				X	G_p	-
73.	TPM_RevokeTrust			X		-	-
74.	TPM_CreateCounter	X				-	U
75.	TPM_ReadCounter				X	-	-
76.	TPM_ReleaseCounter		X			-	U
77.	TPM_ReleaseCounterOwner	X				-	U
78.	TPM_ReadPubek				X	-	-
79.	TPM_DisablePubekRead	X				-	U
80.	TPM_OwnerReadPubek	X				-	U
81.	TPM_MakeIdentity	X				G_p; K_w; S	U
82.	TPM_ActivateIdentity	X				K_u	U
83.	TPM_Delegate_CreateKeyDelegation	X				-	U; K_w
84.	TPM_Delegate_CreateOwnerDelegation	X				-	U; K_w
85.	TPM_Delegate_LoadOwnerDelegation	X				-	U
86.	TPM_Delegate_Manage	X				-	U
87.	TPM_Delegate_ReadTable				X	-	-
88.	TPM_Delegate_UpdateVerification	X				K_u	U
89.	TPM_Delegate_VerifyDelegation	X				K_u	U
90.	TPM_EstablishTransport		X			-	U
91.	TPM_ExecuteTransport		X			K_u	U
92.	TPM_ReleaseTransportSigned		X			S	U
93.	TPM_FlushSpecific				X	-	-
94.	TPM_ForceClear			X		-	-
95.	TPM_GetTicks				X	-	-
96.	TPM_TickStampBlob		X			-	U
97.	TPM_IncrementCounter		X			-	U
98.	TPM_KeyControlOwner	X				-	U
99.	TPM_ResetLockValue	X				-	U
100.	TPM_SetCapability	X				-	U
101.	TPM_SetOperatorAuth			X		-	-
102.	TPM_SetOwnerInstall			X		-	-
103.	TPM_SetOwnerPointer				X	-	-
104.	TPM_changeAuthAsymStart		X			-	-
105.	TPM_changeAuthAsymStart		X			-	-
Locality Controlled Functions							
106.	TPM_HASH_START				X	-	-

Services		Roles				CSPs	
#	Name	CO	User	PP	NoAuth	Private	Secret
107.	TPM_HASH_DATA				X	-	-
108.	TPM_HASH_END				X	-	-
TPM Connection Services							
109.	TSC_PhysicalPresence				X	-	-
110.	TSC_ResetEstablishmentBit				X	-	-
Atmel Specific Services							
111.	TPM_SetState				X	-	-
112.	TPM_OwnerSetState	X				-	U
113.	TPM_GetState				X	-	-
114.	TPM_Identify				X	-	-
115.	TPM_VerifySignature				X	-	-
116.	TPM_BindV20				X		K_w
117.	TPM_DeepSleep (I2C protocol only)				X	-	-
118.	TPM_Lock_FIPS				X	-	-

Note 1: For WIN8 FIPS configuration this command can be run with a NoAuth role.

12 Physical Security Policy

Security Level 1

The TOE meets Physical Security protection requirements for FIPS level 1. Physical security at level 1 assumes no physical protection of CSPs. No actions are required by the operator(s) to ensure that physical security is maintained. Some physical security protection mechanisms beyond the requirements for level 1 have been implemented and are described in the section titled **Mitigation of Other Attacks Policy.**

13 Operational Environment

The FIPS 140-2 level 1 operational environment requirements are not applicable to the TOE. The TOE Operational Environment is non-modifiable. No mechanism exists to upgrade, read, modify or delete the module firmware or hardware. No general purpose operating system is supported or used by the module. Execution of module services is restricted to a single user operating on a single service at any time.

The TOE executable firmware is split into two portions. All executable firmware is stored in protected nonvolatile memory – either ROM or EEPROM. The portion of executable firmware stored in ROM is established in the chip metal layers during the wafer fabrication process. The portion of executable code stored in EEPROM is written and locked into TPM protected EEPROM during the chip production test process. EEPROM executable code cannot be read by any process except through internal execution mechanisms performing explicit execution of valid service requests during the lifetime of the module. EEPROM executable code is stored in memory locations that are protected from modification during the lifetime of the module. Integrity verification of executable firmware is performed separately for ROM and EEPROM code.

Integrity of executable firmware stored in ROM and EEPROM is verified through a SHA-1 hash of the code which is executed during the power on self tests and compared to a value stored in nonvolatile memory when the EEPROM image is loaded and locked into the module. The stored value stored in EEPROM is protected from exposure, modification or deletion. The value cannot be changed for the lifetime of the module.

The TOE does not implement an operating system as defined by FIPS 140-2. Entry of data into the module is controlled by mechanisms for command, status and data communication specified in the **TPM Interface Specification (TCG PC Client Specific TPM Interface Specification (TIS); Specification Version 1.21;**

Revision 1.00; 11 April, 2011; Trusted Computing Group for LPC protocol units and **TPM Interface Specification (TCG PC Client Specific TPM Interface Specification (TIS); Specification Version 1.3; Revision 27; 21 March 2013; Trusted Computing Group** for SPI protocol units. Entry of data in I2C protocol units conform to the **I2C Bus Specification and User Manual, Rev 5, 9 October 2012**. The set of possible functions is limited to the services listed in Table 10 – TOE Services and specified in the TCG TPM Main Specification [2, 3, 4]**.** Any input data or service requests outside those listed in Table 10 will result in failure of the requested service, transmission of a failure code by the TPM and a return to the idle state. Execution is restricted to a single service at any one time. Initiation of any service will cause the module state machine to transition to the service execution state. While the module is in the execution state, requests for new services will be ignored. The module will continue execution until completion, either passing or failing, at which time the module will return a status code and transition to the idle state.

The TOE hardware is built using a proprietary wafer fabrication process and cannot be modified during the lifetime of the module.

13.1 Operating ranges

Normal operating ranges are defined in the respective TOE datasheet [8, 18, 19]. 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.

Temperature: The normal operating temperature range of the TOE is -40°C to +85°C.

Voltage: The normal operating voltage range of the TOE is 3.0V to 3.6V.

Frequency:

LPC units: The normal operating frequency for the LCLK input pin is 32.25 MHz to 33.90 MHz. The minimum clock period is 29.5 nsec. The maximum clock period is 31.0 nsec.

SPI, I2C units: The internal system clock is created by an internal oscillator cell that has no external access.

14 Mitigation of Other Attacks Policy

Security Level 1

The TOE 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.

14.1 Housing

The TOE is housed in an opaque, non-removable, tamper evident, package. Attempts to access the internal IC are detectable by visual inspection of the package.

14.2 Internal Tamper Detection

The TOE 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 FAIL mode.

14.3 Environmental protection

The TOE contains circuitry which will detect environmental conditions outside the range described in the product datasheet. Temperature, power supply voltage and the clock pulse width are 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.

15 References

Reference Number	Reference Title
1	FIPS PUB 140-2, Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST), CHANGE NOTICES (12-03-2002)
2	TPM Main; Part 1 Design Principles; Specification Version 1.2; Level 2; Revision 116; 1 March, 2011; Trusted Computing Group
3	TPM Main; Part 2 Structures; Specification Version 1.2; Level 2; Revision 116; 11 April, 2011; Trusted Computing Group
4	TPM Main; Part 3 Commands; Specification Version 1.2; Level 2; Revision 116; 1 March, 2011; Trusted Computing Group
5	TCG PC Client Specific TPM Interface Specification (TIS); Specification Version 1.21; Revision 1.00; 11 April, 2011; Trusted Computing Group
6	TCG PC Client Specific Implementation Specification for Conventional BIOS; Specification Version 1.21; Revision 1.00; 24 February, 2012 for TPM Family 1.2; Level 2.
7	Trusted Platform Module Atmel-Specific Commands Reference User Guide; AT97SC3204; Atmel Corporation; 300B—TPM—02/08
8	Trusted Platform Module; AT97SC3204; LPC Interface Datasheet 5294IX; 10-2013
9	Trusted Computing Group Physical Presence Interface Specification; Specification version 1.2; Version 1.20; Revision 1.00; February 10, 2011
10	TCG 1.2 TPM Physical Presence Management; Atmel Corporation
11	National Institute of Standards and Technology and Communications Security Establishment, *Derived Test Requirements(DTR) for FIPS PUB 140-2, Security Requirements for Cryptographic Modules*
12	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.
13	National Institute of Standards and Technology, *Annex A: Approved Security Functions for FIPS PUB 140-2, Security Requirements for Cryptographic Modules*
14	National Institute of Standards and Technology, *Secure Hash Standard*, Federal Information Processing Standards Publication 180-4, March, 2012.
15	National Institute of Standards and Technology, *Annex C: Approved Random Number Generators for FIPS 140-2, Security Requirements for Cryptographic Modules*.
16	Intel Low Pin Count (LPC) Interface Specification; Revision 1.1; August, 2002
17	PCI Local Bus Specification; Revision 2.2; December 18, 1998
18	Trusted Platform Module; AT97SC3205; SPI Interface Datasheet 8820GX; 12-2013
19	Trusted Platform Module; AT97SC3205T; 2-Wire Serial Interface Datasheet 8875BX; 12-2013
20	Atmel Specific TPM Commands Reference Guide; AT97SC3205; Application Note 5300EX; 12-2013; Atmel Corporation
21	TCG PC Client Specific TPM Interface Specification (TIS); Specification Version 1.3; Revision 27; 21 March 2013; Trusted Computing Group

22	I2C Bus Specification and User Manual, UM10204; Rev 5, 9 October 2012

16 Definitions and Acronyms

Term	Description
AES	Symmetric cryptographic algorithm. Reference: http://csrc.nist.gov/CryptoToolkit/aes/
AIK	Attestation Identity Key: a special purpose signature key created by the TPM; an asymmetric key, the private portion of which is non-migratable and protected by the TPM. The public portion of an AIK is part of the AIK Credential, issued using either the Privacy CA or DAA protocol. An AIK can only be created by the TPM Owner or a delegate authorized by the TPM Owner. The AIK can be used for platform authentication, platform attestation and certification of keys.
AIK Credential	A credential issued by a Privacy CA that contains the public portion of an AIK key signed by a Privacy CA. The meaning and significance of the fields and the Privacy CA signature is a matter of policy. Typically it states that the public key is associated with a valid TPM.
Attestation	The process of vouching for the accuracy of information. External entities can attest to shielded locations, protected capabilities, and Roots of Trust. A platform can attest to its description of platform characteristics that affect the integrity (trustworthiness) of a platform. Both forms of attestation require reliable evidence of the attesting entity.
Attestation by the TPM	An operation that provides proof of data known to the TPM. This is done by digitally signing specific internal TPM data using an AIK. The acceptance and validity of both the integrity measurements and the AIK itself are determined by the Verifier. The AIK is obtained using either the Privacy CA or DAA protocol.
Attestation of the Platform	An operation that provides proof of a set of the platform’s integrity measurements. This is done by digitally signing a set of PCRs using an AIK in the TPM.
Attestation to the Platform	An operation that provides proof that a platform can be trusted to report integrity measurements; performed using the set or subset of the credentials associated with the platform; used to create an AIK credential.
Authentication of the platform	Provides proof of a claimed platform identity. The claimed identity may or may not be related to the user or any actions performed by the user. Platform Authentication is performed using any non-migratable key (e.g., an AIK). Since there are an unlimited number of non-migratable keys associated with the TPM there are an unlimited number of identities that can be authenticated.
Blob	Generally meaning encrypted data that is generated by a TPM (for use in Protected Storage, or for saving context outside the TPM)
CMK	Certified Migration Key: a key whose migration from a TPM requires an authorization token created with private keys. The corresponding public keys are incorporated in the CMK and referenced when a TPM produces a credential describing the CMK. If a CMK credential is signed by an AIK, an external entity has evidence that a particular key (1) is protected by a valid TPM and (2) requires permission from a

	specific authority before it can be copied.
CRTM	Core RTM: the instructions executed by the platform when it acts as the RTM (Root of Trust for Measurement)
Challenger	(Properly “Identity Challenger”) An entity that requests and has the ability to interpret integrity metrics.
Conformance Credential	A credential that vouches for the conformance of the TPM and the TBB to the TCG specifications
DAA	Direct Anonymous Attestation: a protocol for vouching for an AIK using zero-knowledge-proof technology.
DAA Issuer	A known and recognized entity that interacts with the TPM to install a set of DAA-credentials in the TPM. The DAA issuer provides certification that the holder of such DAA-credentials meets some criteria defined by the Issuer. In many cases the Issuer will be the platform manufacturer, but other entities can become issuers.
Delegation	A process that allows the Owner to delegate a subset of the Owner’s privileges (to perform specific TPM operations).
Denial-of-Service (attack)	An attack which has no affect on information except to prevent its use
Digest	The resulting output of a SHA-1 hash operation
Endorsement Key	EK; an RSA Key pair composed of a public key (EKpu) and private (EKpr). The EK is used to recognize a genuine TPM. The EK is used to decrypt information sent to a TPM in the Privacy CA and DAA protocols, and during the installation of an Owner in the TPM.
Endorsement Key Credential	A credential containing the EKpu that asserts that the holder of the EKpr is a TPM conforming to TCG specifications. Most TPMs are implemented in hardware, but this is not mandatory.
Integrity challenge	A process used to send accurate integrity measurements and PCR values to a challenger.
Integrity Measurement (Metrics)	The process of obtaining metrics of platform characteristics that affect the integrity (trustworthiness) of a platform; storing those metrics; and putting digests of those metrics in shielded locations (called Platform Configuration Registers: PCRs)
Integrity Storage	Storage of integrity metrics in a log and storage of a digest of those metrics in PCRs.
Integrity Reporting	The process of attesting to the contents of integrity storage.
Locality	A mechanism for supporting a privilege hierarchy in the platform
Migratable (key)	A key which is not bound to a specific TPM and with suitable authorization can be used outside a TPM or moved to another TPM.
Non-migratable (key)	A key which is bound to a single TPM; a key that is (statistically) unique to a single TPM but may be moved between TPMs using the maintenance process
Non-volatile	A shielded storage location whose contents are guaranteed to persist

(shielded location)	between uses by Protected Capabilities.
Operator	Anyone who has physical access to a platform
Owner	The entity responsible for the platform’s security and privacy policies, that is distinguished by knowledge of the Owner authorization data.
PCR	Platform Configuration Register: a shielded location containing a digest of integrity digests.
Platform Credential	A credential, typically a digital certificate, attesting that a specific platform contains a unique TPM and TBB.
Protected Capabilities	The set of commands with exclusive permission to access shielded locations
Privacy CA	An entity, typically a Trusted Third Party (TTP), that blinds a verifier to a platform’s EK. An entity (typically well known and recognized) trusted by both the Owner and the Verifier, that will issue AIK Credentials. A Verifier may also adopt the role of a Privacy CA. In that case the roles are co-located but are logically distinct.
Root of Trust (component)	A component that must always behave in the expected manner, because its misbehavior cannot be detected. The complete set of Roots of Trust has at least the minimum set of functions to enable a description of the platform characteristics that affect the trustworthiness of the platform.
RSA	Reference: http://www.rsa.com
RTM	“Root of Trust for Measurement”: a computing engine capable of making inherently reliable integrity measurements. Typically the normal platform computing engine, controlled by the CRTM. This is the root of the chain of transitive trust.
RTS	“Root of Trust for Storage”: a computing engine capable of maintaining an accurate summary of values of integrity digests and the sequence of digests.
RTR	“Root of Trust for Reporting”: a computing engine capable of reliably reporting information held by the RTS.
SHA-1	Reference: http://csrc.ncsl.nist.gov/cryptval/shs.html
Shielded Location	A place (memory, register, etc.) where it is safe to operate on sensitive data; data locations that can be accessed only by “protected capabilities”.
SRK	Storage Root Key: the root key of a hierarchy of keys associated with a TPM’s Protected Storage function; a non-migratable key generated within a TPM.
TCG	Trusted Computing Group
TSS	Trusted Software Stack:software services that facilitate the use of the TPM but do not require the protections afforded to the TPM.
TBB	Trusted Building Block: the parts of the Root of Trust that do not have shielded locations or protected capabilities. Normally includes just the

	instructions for the RTM and the TPM initialization functions (reset, etc.). Typically platform-specific. One example of a TBB is the combination of the CRTM, connection of the CRTM storage to a motherboard, the connection of the TPM to a motherboard, and mechanisms for determining Physical Presence.
Transitive Trust	Also known as "Inductive Trust", in this process the Root of Trust gives a trustworthy description of a second group of functions. Based on this description, an interested entity can determine the trust it is to place in this second group of functions. If the interested entity determines that the trust level of the second group of functions is acceptable, the trust boundary is extended from the Root of Trust to include the second group of functions. In this case, the process can be iterated. The second group of functions can give a trustworthy description of the third group of functions, etc. Transitive trust is used to provide a trustworthy description of platform characteristics, and also to prove that non-migratable keys are non-migratable
Trust	Trust is the expectation that a device will behave in a particular manner for a specific purpose.
TPM	Trusted Platform Module: an implementation of the functions defined in the TCG Trusted Platform Module Specification; the set of Roots of Trust with shielded locations and protected capabilities. Normally includes just the RTS and the RTR.
User	An entity that is making use of the TPM capabilities
Validation Credential	A credential that states values of measurements that should be obtained when measuring a particular part of the platform when the part is functioning as expected.
Verifier	In the DAA model: the entity that interacts with the TPM using the DAA protocol to verify that the TPM has a valid set of DAA-credentials. The verifier may then produce an AIK credential, without reference to the platform EK. In the "Trusted Third Party" model: the entity that requests, receives, and evaluates attestation information based on the EK. The TTP (Privacy CA) may then produce an AIK credential, after verifying the platform EK