# AT97SC3201 Security Target

**Version: 2.3**

**Date: February 21, 2005**

## Interpretations

This ST conforms to the NIAP and International interpretations listed in the following two tables.

### NIAP Interpretations

| # | Title |
|---|---|
| **I-0347** | Including Sensitive Information In Audit Records |
| **I-0350** | Clarification Of Resources/Objects For Residual Information Protection |
| **I-0352** | Rules Governing Binding Should Be Specifiable |
| **I-0375** | Elements Requiring Authentication Mechanism |
| **I-0381** | Relationship Between FPT_PHP And FMT_MOF |
| **I-0389** | Recovery To A Known State |
| **I-0393** | A Completely Evaluated ST Is Not Required When TOE Evaluation Starts |
| **I-0395** | Security Attributes Include Attributes Of Information And Resources |
| **I-0405** | American English Is An Acceptable Refinement |
| **I-0406** | Automated Or Manual Recovery Is Acceptable |
| **I-0407** | Empty Selections Or Assignments |
| **I-0409** | Other Properties In FMT_MSA.3 Should Be Specified By Assignment |
| **I-0410** | Auditing Of Subject Identity For Unsuccessful Logins |
| **I-0411** | Guidance Includes AGD_ADM, AGD_USR, ADO, And ALC_FLR |
| **I-0412** | Configuration Items In The Absence Of Configuration Management |
| **I-0414** | Site-Configurable Prevention Of Audit Loss |
| **I-0415** | User Attributes To Be Bound Should Be Specified |
| **I-0416** | Association Of Access Control Attributes With Subjects And Objects |
| **I-0417** | Association Of Information Flow Attributes W/Subjects And Information |
| **I- 0418** | Evaluation Of The TOE Summary Specification: Part 1 Vs Part 3 |
| **I-0420** | Attribute Inheritance/Modification Rules Need To Be Included In Policy |
| **I-0421** | Application Notes In Protection Profiles Are Informative Only |
| **I-0422** | Clarification Of ``Audit Records'' |
| **I-0423** | Some Modifications To The Audit Trail Are Authorized |
| **I-0424** | FPT_SEP.2 And FPT_SEP.3 Are Not Hierarchical |
| **I-0425** | Settable Failure Limits Are Permitted |
| **I-0426** | Content Of PP Claims Rationale |
| **I-0427** | Identification Of Standards |
| **I-0429** | Selecting One Or More |
| **I-0459** | CM Systems May Have Varying Degrees Of Rigor And Function |

## International Interpretations

| # | Title |
|---|---|
| 003 | Unique identification of configuration items in the configuration list |
| 004 | ACM_SCP.*.1C requirements unclear |
| 006 | Virtual machine description |
| 008 | Augmented and Conformant overlap |
| 009 | Definition of Counter |
| 013 | Multiple SOF claims for multiple domains in a single TOE |
| 016 | Objective for ADO_DEL |
| 019 | Assurance Iterations |
| 024 | COTS product in TOE providing security |
| 025 | Level of detail required for hardware descriptions |
| 027 | Events and actions |
| 031 | Obvious vulnerabilities |
| 032 | Strength of Function Analysis in ASE_TSS |
| 033 | CC use of "Check" |
| 037 | ACM on Product or TOE? |
| 043 | Meaning of "clearly stated" in APE/ASE_OBJ.1 |
| 049 | Threats met by environment |
| 051 | Use of documentation without C & P elements. |
| 055 | Incorrect Component referenced in Part 2 Annexes, FPT_RCV |
| 058 | Confusion over refinement |
| 064 | Apparent higher standard for explicitly stated requirements |
| 065 | No component to call out security function management |
| 067 | Application notes missing |
| 069 | Informal Security Policy Model |
| 074 | Duplicate informative text for ATE_COV.2-3 and ATE_DPT.1-3 |
| 075 | Duplicate informative text for different work units |
| 084 | Aspects of objectives in TOE and environment |
| 085 | SOF Claims additional to the overall claim |
| 095 | SCP Dependency in ACM_CAP |
| 098 | Limitation of refinement |
| 116 | Indistinguishable work units for ADO_DEL |
| 120 | Sampling of process expectations unclear |
| 127 | Work unit not at the right place |
| 128 | Coverage of the delivery procedures |
| 133 | Consistency analysis in AVA_MSU.2 |
| 138 | Iteration and narrowing of scope |

## Revision History

| Version | Date | Description |
|---|---|---|
| 0.1 | 12-Apr-02 | First internal draft |
| 0.2 | 01-May-02 | Draft sent to Atmel, Colorado Springs for comment |
| 1.0 | 09-May-02 | First official release |
| 1.1 | 07-June-02 | Updates to comply with version 1.9.6 of the TCPA TPM PP |
| 1.2 | 28-June-02 | Updated SOF analysis |
| 1.3 | 17-July-03 | Updated section 6. |
| 1.4 | 15-September-03 | Updated ST in response to EORs. |
| 1.5 | 5 November-2003 | Updated ST in response to EORs |
| 1.6 | 2 December 2003 | Updated ST in response to evaluator comments |
| 1.7 | 9 April 2004 | Updated ST in response to EORs |
| 1.8 | 26 April 2004 | Minor corrections |
| 1.9 | 15 June 2004 | Updated ST in response to EORs |
| 2.0 | 6 August 2004 | Updated ST in response to EORs |
| 2.1 | 25 October 2004 | Updated ST in response to validator comments to ETR. |
| 2.2 | 17 November 2004 | Added additional commands available before identification and authentication to FIA_UAU.1 and FIA_UID.1. |
| 2.3 | 21February 2005 | Updated attack potential score in SOF analysis. |

# Contents

Page

## Tables

**Page**

# 1 Introduction

## 1.1 Identification

TOE Identification: Trusted Platform Module Atmel AT97SC3201

ST Identification: Trusted Platform Module Atmel AT97SC3201 Security Target

Version Number: Version 2.3

Date: February 21, 2005

Author: Atmel Corporation

Assurance Level: Evaluation Assurance Level 3 (EAL3) augmented by ADV_SPM.1 and ALC_FLR.1. The strength of function is SOF Medium.

Keywords: Trusted Platform Module, RSA

## 1.2 Security Target Overview

This ST describes the Atmel AT97SC3201, which is an integrated circuit chip designed to be included in personal computers and other embedded systems. The AT97SC3201 implements a Trusted Computing Module (TPM) in accordance with version 1.1b of the TCG Main Specification. The TPM provides security primitives in a secure environment. The primitives include digital signatures, random number generation, and protected storage and binding information to the TPM. The TCG TPM is described in detail in the TCG Main Specification.

## 1.3 Related Documents

- Trusted Computing Group (TCG) Main Specification, version 1.1b.
- TCG Compliance Configuration Specification Version 0.5, August 14, 2001
- International Standard ISO/IEC 15408 Information technology — Security techniques — Evaluation criteria for IT security
- Common Methodology for Information Security Evaluation (CEM) Version 1.0, August 1999
- Common Criteria (CC) Version 2.1 (ISO/IEC 15408 Evaluation Criteria for Information Technology Security; Part 1: Introduction and general model, Part 2: Security functional requirements, and Part 3: Security assurance requirements).
- AT97SC3201 Technical Data Sheet (Atmel Lit. No. 2015)
- Low Pin Count (LPC) Interface Specification, Revision 1.0, September 29, 1997
- Atmel – Specific Commands for TCPA Chip, Version 0.17, 4/12/02

## 1.4 Security Target Organization

The main sections of the ST are the TOE Description, TOE Security Environment, Security Objectives, IT Security Requirements, TOE Summary Specification, PP Claims and Rationale.

Section 2, the TOE Description, provides general information about the TOE, serves as an aid to understanding its security requirements, and provides context for the ST's evaluation.

The TOE Security Environment in Section 3 describes security aspects of the environment in which the TOE is to be used and the manner in which it is to be employed. The TOE security environment includes:

a) Assumptions regarding the TOE's intended usage and environment of use

b) Threats relevant to secure TOE operation

c) Organizational security policies with which the TOE must comply

Section 4 contains the security objectives that reflect the stated intent of the ST. The objectives define how the TOE will counter identified threats and how it will cover identified organizational security policies and assumptions. Each security objective is categorized as being for the TOE or for the environment.

Section 5 contains the applicable Security Requirements taken from the Common Criteria, with appropriate refinements. The requirements are provided in separate subsections for the TOE and its environment. The IT security requirements are subdivided as follows:

a) TOE Security Functional Requirements

b) TOE Security Assurance Requirements

Section 6 contains the TOE Summary Specification.

Section 7 contains the PP Claims.

The Rationale in Section 8 presents evidence that the ST is a complete and cohesive set of requirements and that a conformant TOE would provide an effective set of IT security countermeasures within the security environment. The Rationale is in three main parts. First, a Security Objectives Rationale demonstrates that the stated security objectives are traceable to all of the aspects identified in the TOE security environment and are suitable to cover them. Then, a Security Requirements Rationale demonstrates that the security requirements (TOE and environment) are traceable to the security objectives and are suitable to meet them. Finally, a PP Rationale shows how the assumptions, threats, objectives and requirements in the ST map to those in the PP.

A glossary of acronyms and terms used in the ST is provided in the Appendix.

## 1.5 Common Criteria Conformance

The TOE is

- Part 2 Common Criteria Version 2.1 Extended, and,
- Part 3 Conformant with Common Criteria Version 2.1, augmented with ADV_SPM.1, Informal security policy model and ALC_FLR.1, Basic flaw remediation.

The ST has been built with Common Criteria (CC) Version 2.1 (ISO/IEC 15408 Evaluation Criteria for Information Technology Security; Part 1: Introduction and general model, Part 2: Security functional requirements, and Part 3: Security assurance requirements).

The ST is conformant with Common Criteria Version 2.1, Part 2 Extended, and Part 3 (Evaluation Assurance Level 3 with augmentation).

This ST conforms to the NIAP and International interpretations listed in the front matter of this document.

# 2 TOE Description

## 2.1 Overview

The TOE is the Atmel AT97SC3201, which is an integrated circuit chip security module designed to be integrated into personal computers and other embedded systems. The AT97SC3201 implements a TPM in accordance with version 1.1b of the TCG Main Specification.

## 2.2 Definition of the TOE as a TPM

The TPM is a collection of hardware, firmware and/or software that supports the following protocols and algorithms:

- Algorithms: RSA, SHA-1, HMAC
- Random number generation
- Key generation
- Self-tests

The TPM may be used to provide secure storage for a minimum of 10 private keys or other data by using RSA key technology to encrypt data and keys. The resulting encrypted file, which contains header information in addition to the data or key, is called a blob and is output by the TPM and can be loaded in the TPM when needed. The functionality of the TPM can also be used so that private keys generated on the TPM can be stored outside the TPM (encrypted) in a way that allows the TPM to use them later without ever exposing such keys in the clear outside the TPM.

The functionality used to provide secure storage is:

- Seal and Unseal, which perform RSA encrypt and decrypt, respectively, on data that is externally generated. The sealing operation encrypts not only the data, but also the platform configuration values that are stored in the platform configuration registers (PCRs) in the TPM and TPMProof, which is a unique identifier for that TPM. To unseal the data, three conditions must exist: 1) the appropriate key must be available for unseal, 2) the TPM PCRs must contain the same values that existed at the time of the seal operation, and 3) the value of TPMProof must be the same as that encrypted during the seal operation. By requiring the PCR values to be duplicated at unseal and the TPMProof value to be checked, the seal operation allows software to explicitly state the future "trusted" configuration that the platform must be in for the decrypted key to be used and for decrypt to only occur on the specified TPM.
- Unbind, which decrypts a blob created outside the TPM that has been RSA encrypted using a public key where the associated private key is stored in the TPM.

The TPM provides evidence of origin when commands that execute a sign operation and transmit signed data are used, including: TPM_Sign, TPM_GetAuditEventSigned and TPM_GetCapabilitySigned. These commands enforce generation of evidence of origin.

A number of key types are defined within the TPM. Keys may be migratable or non-migratable. A migratable key is a key that may be transported outside the specific TPM. A non-migratable key is a key that cannot be transported outside a specific TPM. Key types include:

- The Storage Root key (SRK), which is the root key of a hierarchy of keys associated with a TPM; it is generated within a TPM and is a non-migratable key. Each TPM contains a SRK, generated by the TPM at the request of the Owner. Under that SRK are two trees: one dealing with migratable data and the other dealing with non-migratable data
- Signing Keys, which must be a leaf of the Storage Root Key hierarchy. The private key of the key pair is used for signing operations only.

- Storage keys, which are used only to RSA encrypt and RSA decrypt other keys in the Protected Storage hierarchy,.
- Identity Keys, which are only used for operations that require a TPM identity.
- Binding Keys, which are used for TPM_Unbind operations only. A bind operation (performed outside the TPM) associates identification and authentication data with a particular data set and the entire data blob is encrypted outside the TPM using a binding key, which is an RSA key. The TPM_Unbind operation uses a private key stored in the TPM to decrypt the blob so that the data (often a key pair) stored in the blob may be used.
- The Endorsement key pair, which is an asymmetric key pair generated by or inserted in a TPM that is used as proof that a TPM is a genuine TPM.

Each TPM is identified and validated by its Endorsement Key. A TPM has only one endorsement key pair. The Endorsement Key is transitively bound to the Platform via the TPM as follows:

1. An Endorsement Key is bound to one and only one TPM (i.e., that is a one to one correspondence between an Endorsement Key and a TPM.)
2. A TPM is bound to one and only one Platform, (i.e., there is a one to one correspondence between a TPM and a Platform.)
3. Therefore, an Endorsement Key is bound to a Platform, (i.e., there is a one to one correspondence between an Endorsement Key and a Platform.

TPM algorithms, protocols, identification and authentication, and access control functions are described in the following subsections.

### 2.2.1 Algorithms

The TPM supports the RSA algorithm and uses the RSA algorithm for encryption and digital signatures. The TPM supports RSA key sizes of 512, 1024, and 2048 bits. The RSA public exponent is e, where $e = 2^{16}+1$. All TPM Storage keys are of strength equivalent to a 2048-bit RSA key or greater. The TPM does not load a Storage key whose strength is less than that of a 2048-bit RSA key. All TPM identity keys are of a strength equivalent to a 2048-bit RSA key or greater.

The TPM supports the Secure Hash Algorithm 1 (SHA-1) as defined by United States Federal Information Processing Standard 180-1. The output of SHA-1 is 160 bits and all areas that expect a hash value support the full 160 bits. An SHA-1 digest is used in the early stages of a boot process, before more sophisticated computing resources are available. Secure Hash is also used in the process of preparing data for signature or signature verification.

The TPM uses the RSA algorithm for signature and verification operations. The TPM uses PKCS #1 V2 for the format and design of the signature output.

Key destruction is accomplished by marking memory locations containing the actual key data with a tag indicating “free memory space.”

### 2.2.2 Random Number Generator (RNG)

The RNG capability is accessible only to valid TPM commands; intermediate results from the RNG are not available to any user. When the data is for internal use by the TPM (e.g., asymmetric key generation), the data is held in a shielded location and is not accessible to any user. For more information on “shielded locations” see Section 3.1 of the TCG Main Specification.

### 2.2.3 Key Generation

The TPM generates asymmetric key pairs. The generate function is a protected capability and the private key is held in a shielded location. For more information on the definition of “protected “ capabilities and “shielded locations” see Section 3.1 of the TCG Main Specification.

### 2.2.4 Self-tests

The TPM provides startup self-tests and a mechanism to allow the self-tests to be run on demand, i.e., the TPM_ContinueSelfTest command. The response from the self-tests is pass or fail. When the TPM detects a failure during any self-test, the part experiencing the failure enters a shutdown mode and an error code is returned.

### 2.2.5 Identification and Authentication

The TPM identification and authentication capability is used to authenticate an entity owner and to authorize use of an entity. The basic premise is to prove knowledge of a shared secret. This shared secret is the identification and authentication data. The TCG Specification calls the identification and authentication process and this data "authorization."

The identification and authentication data for the TPM Owner and the owner of the Storage Root Key are held within the TPM itself. The identification and authentication data for other owners of entities are held and protected with the entity.

The identification and authentication protocols use a random nonce. This requires that a nonce from one side be in use only for a message and its reply. For instance, the TPM would create a nonce and send that on a reply. The requestor would receive that nonce and then include it in the next request. The TPM would validate that the correct nonce was in the request and then create a new nonce for the reply. This mechanism is in place to prevent replay attacks.

### 2.2.6 Access Control

Access control is enforced in the TPM on all data and operations performed on that data. The TPM provides access control by denying access to some data and operations and allowing access to other data and operations based on the value of the TCPA_AUTH_DATA_USAGE flag, TCPA_KEY_FLAGS, and the TCPA_KEY_USAGE flag. The TCPA_AUTH_DATA_USAGE flag defines access as either owner or world. Owner must be authenticated with a shared secret as described in Section 2.2.5, above. World means that usage of the key is permitted by anyone without authentication. The TCPA_KEY_FLAGS define whether a key is migratable or non-migratable and whether the key is stored in volatile storage and must be unloaded at TPM startup. The TCPA_KEY_USAGE flag identifies the key type, as defined in Section 2.2, above. Depending on the key type, certain operations may or may not be allowed using the particular key, as described above.

Upon appropriate identification and authentication associated with the keys, users can use the key for the purposes permitted by the TCPA_KEY_USAGE flag.

### 2.2.7 Security Management and Protection of the TOE

The TPM provides security management functionality and functions that protect the TOE. These functions include:

- Restricting the ability to disable or enable security functions
- Management of security attributes associated with keys or data through a set of flags
- Enforcing specific default values for security attributes, with override capability of those default values limited to the entity owner
- Enforcement of roles, including TPM owner, entity owner, and manufacturer (or designee)
- Preservation of secure state for specific types of failures
- Function recovery to a secure state with the return of an error code
- Replay detection

- Maintaining a security domain for its own execution and not allowing any functions to bypass authorization command sequences
- Consistent interpretation of commands through the restriction of possible commands to only those specified in the design documentation
- Trusted path

## 2.3 Security Attributes and Data

All data, including user key pairs, user data, and TSF data, have associated security attributes, stored as flags in the TPM or associated with the data in an encrypted blob. The following security attributes are defined:

- Migration attribute, which determines if the data (or key pair) can migrate from one TPM to another. This security attribute is stored in TCPA_KEY_FLAGS.
- TCPA_AUTHDATA_USAGE flag is used to define whether the data can be accessed only by the owner or by the world.
- Attribute key type, stored in TCPA_ KEY_USAGE, which indicates if the data is a key or key pair and the type of key (e.g., storage, binding, etc.).
- Volatility attribute, which defines whether the data must be stored in volatile or non-volatile storage and if it is cleared at TPM startup. This security attribute is stored in TCPA_KEY_FLAGS.

Within the TPM, for the purposes of Common Criteria evaluation, TSF data is defined as:

- The Endorsement Key Pair,
- The Storage Root Key (SRK),
- TPMProof, i.e., the random number (nonce) that each TPM maintains to validate that the data originated at this TPM.
- PCR values,
- TPM owner identification and authentication data,
- Entity owner identification and authentication data,
- Migration authorization data, which is used in creating migratable key blobs,
- Security attributes as defined above.

User data is defined as all user keys and other data that may be passed to the TPM for signature, decryption, etc.

## 2.4 TOE Boundary

The TOE comprises the Atmel AT97SC3201 and its embedded firmware. The TOE performs RSA key generation and digital signature, data decryption, user identification and authentication, secure hash, and software random number generation. The TSF boundary is the same as the TOE boundary.

## 2.5 TOE Environment

The TOE is designed to be integrated into personal computers and other embedded systems. All communication between the host system and the TOE is through the LPC interface on the TOE.

The TOE is offered to OEM manufacturers as a turnkey solution, including the embedded firmware. In addition, Atmel provides the necessary driver software for integration into certain operating systems, along with BIOS drivers.

Operation of the TOE is possible only after initialization of the TOE at the user site. Initialization is not performed at the factory.

# 3 TOE Security Environment

This section identifies the following:

- Secure usage assumptions,
- Organizational security policies, and
- Threats to Security

## 3.1 Secure Usage Assumptions

TOE secure usage assumptions are defined in Table 1.

*Table 1. Secure Usage Assumptions*

| # | Assumption | Description |
|---|---|---|
| 1 | A.Configuration | It is assumed that the TOE will be properly installed and configured. |

## 3.2 Threats to Security

Threats to the TOE are defined in Table 2. The asset under attack is the information stored in or moving in or out of the TOE. In general, the threat agent includes, but is not limited to: 1) people with TOE access who are expected to possess “average” expertise, few resources, and moderate motivation, or 2) failure of the TOE.

*Table 2. Threats to Security*

| # | Threat | Description |
|---|---|---|
| 1 | T.Attack | An undetected compromise of the cryptography-related IT assets may occur as a result of an authorized or unauthorized user attempting to perform actions that the user is not authorized to perform. |
| 2 | T.Bypass | An unauthorized individual or user may tamper with security attributes or other data in order to bypass TOE security functions and gain unauthorized access to TOE assets. |
| 3 | T.Export | An authorized or unauthorized user may export data without security attributes or with unsecured security attributes, causing the data exported to be erroneous and unusable, to allow erroneous data to be added or substituted for the original data, and/or to reveal secrets. |
| 4 | T.Hack_Crypto | Cryptographic algorithms may be incorrectly implemented, allowing an unauthorized user or authorized user to decipher keys generated within the TPM and thereby gain unauthorized access to encrypted data. |
| 5 | T.I&A | An authorized or unauthorized TOE user may gain access to TOE data, keys, and operations to which they are not authorized access. |
| 6 | T.Import | An authorized or unauthorized user may import data or keys without security attributes or with erroneous security attributes, causing key ownership and authorization to be uncertain or erroneous and the system to malfunction or operate in an insecure manner. |
| 7 | T.Key_Gen_Destroy | Cryptographic keys may be generated or destroyed by an authorized user in an insecure manner, causing key compromise. |
| 8 | T.Malfunction | TOE assets may be modified or disclosed to an authorized or unauthorized user of the TOE, through malfunction of the TOE. |

| # | Threat | Description |
|---|---|---|
| 9 | T.Modify | An unauthorized user may modify TSF or user data, e.g., stored security attributes or keys, in order to gain access to the TOE and its assets. |
| 10 | T.Object_Attr_Default | An authorized user may create an object with no security attribute values, causing a sensitive object to be accessible to unauthorized users. |
| 11 | T.Object_Attr_Change | An authorized or unauthorized user may make unauthorized changes to security attribute values for an object, causing a sensitive object to be accessible to unauthorized users. |
| 12 | T.Object_SecureValues | An authorized user may set unsecured values for object security attributes, causing a sensitive object to be accessible to unauthorized users. |
| 13 | T.Residual_Info | An authorized user may obtain information that the user is not authorized to have when the data is no longer actively managed by the TOE ("data scavenging") causing keys or other sensitive data to be compromised. |
| 14 | T.Replay | An unauthorized user may gain access to the system and sensitive data through a "replay" attack that allows the user to capture identification and authentication data. |
| 15 | T.Repudiate_Transact | An originator of data may deny originating the data to avoid accountability. |
| 16 | T.Test | The TOE may start up in an insecure state or enter an insecure state, allowing an authorized or unauthorized user to obtain sensitive data or compromise the system. |

# 4 Security Objectives

## 4.1 Security Objectives for the TOE

TOE security objectives are defined in Table 3.

*Table 3. Security Objectives for the TOE*

| # | Objective | Description |
|---|---|---|
| 1 | O.Crypto_Key_Man | The TOE shall generate and destroy cryptographic keys in a secure manner. |
| 2 | O.Crypto_Op | The TOE shall perform cryptographic operations, including secure hash, random number generation, HMAC, RSA digital signature and signature verification, RSA encryption and decryption, and RSA key generation in accordance with specified algorithms and key size; key size must be sufficient size to minimize the risk of deciphering private/public key pairs. |
| 3 | O.Self_Test | The TOE shall provide the ability to verify that the TSF functions operate as designed. |
| 4 | O.DAC | The TOE shall control and restrict user access to the TOE assets in accordance with a specified access control policy. |
| 5 | O.Export | When data are exported outside the TPM, the TOE shall ensure that the data security attributes being exported are unambiguously associated with the data. |
| 6 | O.Fail_Secure | The TOE shall preserve the secure state of the system in the event of a cryptographic or other failure. |
| 7 | O.General_Integ_Checks | The TOE shall provide periodic checks on system integrity and user data integrity. |
| 8 | O.HMAC | The TOE shall provide the ability to detect the modification of security attributes and other data. |
| 9 | O.I&A | The TOE shall uniquely identify all users, and shall authenticate the claimed identify before granting a user access to the TOE facilities that require authorization. |
| 10 | O.Import | When data are being imported into the TOE, the TOE shall ensure that the data security attributes are being imported with the data and the data is from authorized source. In addition, the TOE shall verify those security attributes according to the TSF access control rules. |
| 11 | O.Invoke | The TSF shall be invoked for all actions. |
| 12 | O.Limit_Actions_Auth | The TOE shall restrict the actions a user may perform before the TOE verifies the identity of the user. |
| 13 | O.MessageNR | The TOE shall provide user data integrity, source authentication, and the basis for source non-repudiation when exchanging data with a remote system. |
| 14 | O.No_Residual_Info | The TOE shall ensure there is no “object reuse,” i.e., ensure that there is no residual information in information containers or system resources upon their reallocation to different users. |
| 15 | O.Object_Attr_Default | The TOE shall require default security attributes for the object when an object is created. |
| 16 | O.Object_Attr_DefaultOver | The TOE shall permit authorized users to override defaulted values for security attributes for an object. |

| # | Objective | Description |
|---|---|---|
| 17 | O.Obj_Attr_SecureValues | The TOE shall maintain object security attributes by permitting only secure values; secure values are security parameters associated with a key that require owner authorization. |
| 18 | O.Security_Attr_Mgt | The TOE shall allow only authorized users to initialize and change object security attributes. |
| 19 | O.Security_Roles | The TOE shall maintain security-relevant roles and association of users with those roles. |
| 20 | O.Self_Protect | The TSF will maintain a domain for its own execution that protects it and its resources from external interference, tampering, or unauthorized disclosure. |
| 21 | O.Single_Auth | The TOE shall provide a single use authentication mechanism and require re-authentication to prevent “replay” attacks. |

## 4.2 Security Objectives for the Environment

Table 4 lists security objectives for the environment.

*Table 4. Security Objectives for the Environment*

| # | Objective Name | Objective Description |
|---|---|---|
| 1 | OE.Configuration | The TOE shall be installed and configured properly for starting up the TOE in a secure state. |

# 5 IT Security Requirements

## 5.1 Introduction

This section defines the TOE security functional requirements and assurance requirements. All requirements are from the CC Parts 2 and 3. Selections, assignments, and refinements are indicated by *italics*.

## 5.2 TOE Security Functional Requirements

This section defines the TOE security functional requirements (SFRs). The full text of the security functional requirements is contained below. The TOE SFRs are Part 2 Extended. The SFRs are Part 2 Extended because there are explicitly stated requirements as well as requirements drawn from Part 2.

The definition of Part 2 extended is found in the CC Part 3, section 5.4, “Part 2 extended - A PP or TOE is Part 2 extended if the functional requirements include functional components not in Part 2. All TOE functional requirements in this ST are listed in Table 5, below. Part 2 extended requirements are explicitly identified as “Part 2 extended.”

For the Part 2 functional requirements the standard CC text is in regular font; the text inserted by the Security Target (ST) author is in *italic* font. Certain security functional requirements have multiple iterations in the text. Iterations are indicated by the use of a “:” in the component identification and by a “;” in the component name.

*Table 5. TOE Security Functional Requirements*

| # | Functional Requirement | Title | Part 2 or Part 2 Extended |
|---|---|---|---|
| 1 | FCO_NRO.2 | Enforced proof of origin | Part 2 |
| 2 | FCS_CKM.1 | Cryptographic key generation | Part 2 |
| 3 | FCS_CKM.4 | Cryptographic key destruction | Part 2 |
| 4 | FCS_COP.1 | Cryptographic operation | Part 2 |
| 5 | FDP_ACC.1 | Subset access control | Part 2 |
| 6 | FDP_ACF.1 | Security attribute based access control | Part 2 |
| 7 | FDP_ETC.2 | Export of user data with security attributes | Part 2 |
| 8 | FDP_ITC.2 | Import of user data with security attributes | Part 2 |
| 9 | FDP_RIP.2 | Full residual information protection | Part 2 |
| 10 | FIA_ATD.1 | User attribute definition | Part 2 |
| 11 | FIA_UAU.1 | Timing of authentication | Part 2 |
| 12 | FIA_UAU.4 | Single-use authentication mechanism | Part 2 |
| 13 | FIA_UAU.6 | Re-authenticating | Part 2 |
| 14 | FIA_UID.1 | Timing of identification | Part 2 |
| 15 | FMT_MOF.1 | Management of security functions behavior | Part 2 |
| 16 | FMT_MSA.1 | Management of security attributes | Part 2 |
| 17 | FMT_MSA.2 | Secure security attributes | Part 2 |
| 18 | FMT_MSA.3 | Static attribute initialization | Part 2 |
| 19 | FMT_MTD.1 | Management of TSF data | Part 2 |

| # | Functional Requirement | Title | Part 2 or Part 2 Extended |
|---|---|---|---|
| 20 | FMT_SMR.2 | Restrictions on security roles | Part 2 |
| 21 | FPT_AMT.1 | Abstract machine testing | Part 2 |
| 22 | FPT_FLS.1 | Failure with preservation of secure state | Part 2 |
| 23 | FPT_RCV.4 | Function recovery | Part 2 |
| 24 | FPT_RPL.1 | Replay detection | Part 2 |
| 25 | FPT_RVM.1 | Non-bypassability of the TSP | Part 2 |
| 26 | FPT_SEP.1 | TSF domain separation | Part 2 |
| 27 | FPT_TDC.1 | Inter-TSF basic TSF data consistency | Part 2 |
| 28 | FPT_TPMTST.1 | TPM integrity test | Part 2 Extended |
| 29 | FTP_TRP.1 | Trusted path | Part 2 |

### 5.2.1 Class FCO – Communication

**FCO_NRO.2 Enforced proof of origin**

| | |
|---|---|
| Hierarchical to: | FCO_NRO.1 |
| FCO_NRO.2.1 | The TSF shall enforce the generation of evidence of origin for transmitted *TPM data signed using identity keys* at all times. |
| FCO_NRO.2.2 | The TSF shall be able to relate the *identity* of the originator of the information, and the *TPM data* of the information to which the evidence applies. |
| FCO_NRO.2.3 | The TSF shall provide a capability to verify the evidence of origin of information to *recipient* given *evidence only available when requestor properly authenticates*. |
| Dependencies: | FIA_UID.1 Timing of identification |

### 5.2.2 Class FCS – Cryptographic Support

**FCS_CKM.1 Cryptographic key generation**

| | |
|---|---|
| Hierarchical to: | No other components. |
| FCS_CKM.1.1 | The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm *RSA* and specified cryptographic key sizes *RSA 512, 1024, 2048* that meet the following: *PKCS#1 V2*. |
| Dependencies: | FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes |

**FCS_CKM.4 Cryptographic key destruction**

| | |
|---|---|
| Hierarchical to: | No other components. |
| FCS_CKM.4.1 | The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *erasure of register bits in TCPA_KEY_FLAGS that indicate the valid/invalid status of cryptographic keys. Memory locations containing the actual key data are marked with a tag indicating "free memory space"* that meets the following: *list of standards - none.* |
| Dependencies: | FCS_CKM.1 Cryptographic key generation, FMT_MSA.2 Secure security attributes |

**FCS_COP.1:1 Cryptographic operation; RSA encrypt and decrypt**

| | |
|---|---|
| Hierarchical to: | No other components. |
| FCS_COP.1.1;1 | The TSF shall perform *encryption and decryption* in accordance with a specified cryptographic algorithm *RSA* and cryptographic key sizes *RSA 512, 1024, 2048* that meet the following: *PKCS#1 V2*. |
| Dependencies: | FCS_CKM.1 Cryptographic key generation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes |

**FCS_COP.1:2 Cryptographic operation; RSA signature and signature verification**

| | |
|---|---|
| Hierarchical to: | No other components. |
| FCS_COP.1.1;2 | The TSF shall perform *signature generation and signature verification* in accordance with a specified cryptographic algorithm *RSA* and cryptographic key sizes *RSA 512, 1024, 2048* that meet the following: *PKCS#1 V2*. |
| Dependencies: | FCS_CKM.1 Cryptographic key generation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes |

**FCS_COP.1:3 Cryptographic operation; SHA**

| | |
|---|---|
| Hierarchical to: | No other components. |
| FCS_COP.1.1; 3 | The TSF shall perform *secure hash* in accordance with a specified cryptographic algorithm *SHA-1* and cryptographic key sizes *not applicable* that meet the following: *FIPS 180-1*. |
| Dependencies: | FCS_CKM.1 Cryptographic key generation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes |

**FCS_COP.1:4 Cryptographic operation; Keyed-Hashing for Message Authentication**

Hierarchical to: No other components.

FCS_COP.1.1; 4 The TSF shall perform *keyed-hashing message authentication code (HMAC)* in accordance with a specified cryptographic algorithm *SHA-1* and cryptographic key sizes *160 bits* that meet the following: *RFC 2104*.

Dependencies: FCS_CKM.1 Cryptographic key generation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes

**FCS_COP.1:5 Cryptographic operation; RNG**

Hierarchical to: No other components.

FCS_COP.1.1;5 The TSF shall perform *random number generation* in accordance with a specified cryptographic algorithm *random number generator* and cryptographic key sizes *not applicable* that meet the following: *TCG Main Specification version 1.1b, Section 10.5*.

Dependencies: FCS_CKM.1 Cryptographic key generation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

### 5.2.3 Class FDP – User Data Protection

**FDP_ACC.1 Subset access control**

Hierarchical to: No other components.

FDP_ACC.1.1 The TSF shall enforce the *Protected Operations Access Controls* on

*a) Subjects: commands executing on behalf of users.*

*b) Objects: keys and user data.*

*c) Operations: signature generation, encryption, or decryption.*

Dependencies: FDP_ACF.1 Security attribute based access control

**FDP_ACF.1 Security attribute based access control**

Hierarchical to: No other components

FDP_ACF.1.1 The TSF shall enforce the *Protected Operations Access Controls* to objects based on *security attributes TCPA_AUTH_DATA_USAGE, TCPA_KEY_FLAGS and TCPA_KEY_USAGE*.

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

*a) Key and data access is defined as "owner" access or "world" based on the value of TCPA_AUTH_DATA_USAGE*

|  |  |
|---|---|
|  | *b) Cryptographic operations for each key are limited based on the specification of the TCPA_KEY_USAGE value.* |
| FDP_ACF.1.3 | The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [*None*]. |
| FDP_ACF.1.4 | The TSF shall explicitly deny access of subjects to objects based on: [*None*]. |
| Dependencies: | FDP_ACC.1 Subset access control, FMT_MSA.3 Static attribute initialization |

**FDP_ETC.2 Export of user data with security attributes**

|  |  |
|---|---|
| Hierarchical to: | No other components |
| FDP_ETC.2.1 | The TSF shall enforce the *Protected Operations Access Controls* when exporting user data, controlled under the SFP, outside of the TSC. |
| FDP_ETC.2.2 | The TSF shall export the user data with the user data's associated security attributes. |
| FDP_ETC.2.3 | The TSF shall ensure that the security attributes, when exported outside the TSC, are unambiguously associated with the exported user data. |
| FDP_ETC.2.4 | The TSF shall enforce the following rules when user data is exported from the TSC: *A key may be encrypted for migration only if the migratable flag is set in TCPA_KEY_FLAGS,* [*no additional exportation control rules*]. |
| *Application note:* | *Security attributes are encrypted in a blob prior to export. As part of the blob that has been encrypted, the security attributes are unambiguously associated with the data.* |
| Dependencies: | FDP_ACC.1 Subset access control |

**FDP_ITC.2 Import of user data with security attributes**

|  |  |
|---|---|
| Hierarchical to: | No other components |
| FDP_ITC.2.1 | The TSF shall enforce the *Protected Operations Access Controls* when importing user data, controlled under the SFP, from outside of the TSC. |
| FDP_ITC.2.2 | The TSF shall use the security attributes associated with the imported user data. |
| FDP_ITC.2.3 | The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received. |
| FDP_ITC.2.4 | The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data. |
| FDP_ITC.2.5 | The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: [*no additional importation control rules*]. |

*Application note:* *Security attributes are imported with data as part of the encrypted blob. As part of the blob that has been encrypted, the security attributes are unambiguously associated with the data.*

Dependencies: FDP_ACC.1 Subset access control, FTP_TRP.1 Trusted path, FPT_TDC.1 Inter-TSF basic TSF data consistency.

**FDP_RIP.2 Full residual information protection**

Hierarchical to: FDP_RIP.1

FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the *de-allocation of the resource from* all objects.

Dependencies: None.

## 5.2.4 Class FIA – Identification and Authentication

*Application note:* *The TPM identification and authentication capability is used to authenticate an entity owner and to authorize use of an entity. The basic premise is to prove knowledge of a shared secret. This shared secret is the identification and authentication data. Note that the TCG Main Specification document refers to the identification and authentication process and this data as authorization.*

**FIA_ATD.1 User attribute definition**

Hierarchical to: No other components

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: *authentication data, role.*

Dependencies: None

**FIA_UAU.1 Timing of authentication**

Hierarchical to: No other components

FIA_UAU.1.1 The TSF shall allow *access to data and keys where the entity owner has given the "world" access based on the value of TCPA_AUTH_DATA_USAGE and the TSF shall allow actions consisting of the execution of the following commands: TPM_SelfTestFull, TPM_ContinueSelfTest, TPM_GetTestResult, TPM_PcrRead, TPM_DirRead, TPM_EvictKey, TPM_DisableForceClear, TPM_CreateEndorsementKeyPair, TPM_Extend, TPM_GetCapability, TPM_GetOrdinalAuditStatus, TPM_OIAP, TPM_ OSAP, TPM_ReadPubek, TPM_Reset, TPM_SaveState, TPM_SetOwnerInstall, TPM_GetRandom, TPM_SetTempDeactivated, TPM_SHA1Complete, TPM_SHA1CompleteExtend, TPM_SHA1Start, TPM_SHA1Update, TPM_Startup, TPMStirRandom, TPM_TerminateHandle*, TPM_SetState,

| | |
|---|---|
| | TPM_GetState, TPM_Identify, TPM_VerifySignature, and TPM_BindV20 on behalf of the user to be performed before the user is authenticated. |
| FIA_UAU.1.2 | The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user. |
| Dependencies: | FIA_UID.1 Timing of identification |

**FIA_UAU.4 Single-use authentication mechanisms**

| | |
|---|---|
| Hierarchical to: | No other components |
| FIA_UAU.4.1 | The TSF shall prevent reuse of authentication data related to *the use of the "Object-Independent Authorization Protocol" (OI-AP) and the "Object-Specific Authorization Protocol" (OS-AP) protocols.* |
| Dependencies: | None. |

**FIA_UAU.6 Re authenticating**

| | |
|---|---|
| Hierarchical to: | No other components |
| FIA_UAU.6.1 | The TSF shall re-authenticate the user under the conditions: *for every command that requires user authentication*. |
| Dependencies: | None. |

**FIA_UID.1  Timing of identification**

Hierarchical to: No other components

| | |
|---|---|
| FIA_UID.1.1 | The TSF shall allow *access to data and keys where the entity owner has given the "world" access based on the value of TCPA_AUTH_DATA_USAGE and the TSF shall allow actions consisting of the execution of the following commands: TPM_SelfTestFull, TPM_ContinueSelfTest, TPM_GetTestResult, TPM_PcrRead, TPM_DirRead, TPM_EvictKey, TPM_DisableForceClear, TPM_CreateEndorsementKeyPair, TPM_Extend, TPM_GetCapability, TPM_GetOrdinalAuditStatus, TPM_OIAP, TPM_OSAP, TPM_ReadPubek, TPM_Reset, TPM_SaveState, TPM_SetOwnerInstall, TPM_GetRandom, TPM_SetTempDeactivated, TPM_SHA1Complete, TPM_SHA1CompleteExtend, TPM_SHA1Start, TPM_SHA1Update, TPM_Startup, TPMStirRandom, TPM_TerminateHandle*, TPM_SetState, TPM_GetState, TPM_Identify, TPM_VerifySignature, and TPM_BindV20 on behalf of the user to be performed before the user is identified. |
| FIA_UID.1.2 | The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user. |
| Dependencies: | None. |

|  |  |
|---|---|
| *Application Note:* | *Identification within the TOE is identification of either the TPM owner or the owner of a specific key pair (also known as an "entity"); identification is performed by verifying the password associated with the TPM owner or the key pair.* |

### 5.2.5 Class FMT – Security Management

**FMT_MOF.1 Management of security functions behavior**

Hierarchical to: No other components

FMT_MOF.1.1 The TSF shall restrict the ability to *disable or enable* the functions to

- *Reset the Authorization Failure Counter to 0x0000 (TPM_FAILCOUNT)*
- *Change the operating mode of the Authorization Failure Counter (TPM_FAILMOD)*
- *Initialize and lock the FIPS operating mode (TPM_FIPS)*

to *the TPM owner*.

Dependencies: FMT_SMR.1 Security roles (met by FMT_SRM.2)

**FMT_MSA.1 Management of security attributes**

Hierarchical to: No other components

FMT_MSA.1.1 The TSF shall enforce the *Protected Operations Access Controls* to restrict the ability to *create* the security attributes *associated with a particular entity, including TCPA_KEY_USAGE, TCPA_AUTH_DATA_USAGE, migratable flag, and volatility flag* to *the entity owner.*

Dependencies: FMT_SMR.1 Security roles (met by FMT_SMR.2), FDP_ACC.1 Subset access control

**FMT_MSA.2 Secure security attributes**

Hierarchical to: No other components

FMT_MSA.2.1 The TSF shall ensure that only secure values are accepted for security attributes.

Dependencies: ADV_SPM.1 Informal TOE security policy model, FMT_SMR.1 Security roles (met by FMT_SMR.2), FDP_ACC.1 Subset access control, FMT_MSA.1 Management of security attributes

**FMT_MSA.3 Static attribute initialization**

Hierarchical to: No other components

| | |
|---|---|
| FMT_MSA.3.1 | The TSF shall enforce the *Protected Operations Access Controls* to provide *specific* default values for security attributes that are used to enforce the SFP. |
| FMT_MSA.3.2 | The TSF shall allow the *entity owner* to specify alternative initial values to override the default values when an object or information is created. |
| Dependencies: | FMT_SMR.1 Security roles (met by FMT_SMR.2), FMT_MSA.1 Management of security attributes |

**FMT_MTD.1:1 Management of TSF data – TPM Owner modify**

Hierarchical to: No other components

| | |
|---|---|
| FMT_MTD.1.1;1 | The TSF shall restrict the ability to *modify* the *TSF data: Identification and Authentication data associated with the Endorsement Key and SRK; Migration authorization data* to *the TPM Owner*. |
| Dependencies: | FMT_SMR.1 Security roles (met by FMT_SMR.2) |

**FMT_MTD.1:2 Management of TSF data – TPM Owner create**

Hierarchical to: No other components

| | |
|---|---|
| FMT_MTD.1.1;2 | The TSF shall restrict the ability to *generate* the *TSF data: Storage Root Key and TPMProof* to *the TPM Owner*. |
| Dependencies: | FMT_SMR.1 Security roles (met by FMT_SMR.2) |

**FMT_MTD.1:3 Management of TSF data – Entity Owner**

Hierarchical to: No other components

| | |
|---|---|
| FMT_MTD.1.1;3 | The TSF shall restrict the ability to *modify* the *TSF data: Identification and Authentication data associated with entity* to *the entity owner*. |
| Dependencies: | FMT_SMR.1 Security roles (met by FMT_SMR.2) |

**FMT_MTD.1:4 Management of TSF data – Manufacturer**

Hierarchical to: No other components

| | |
|---|---|
| FMT_MTD.1.1;4 | The TSF shall restrict the ability to *generate* the *TSF data: Endorsement Key Pair* to *the TPM manufacturer or designee*. |
| Dependencies: | FMT_SMR.1 Security roles (met by FMT_SMR.2) |

**FMT_SMR.2 Restrictions on security roles**

| | |
|---|---|
| Hierarchical to: | FMT_SMR.1 |

| | |
|---|---|
| FMT_SMR.2.1 | The TSF shall maintain the roles: *TPM Owner, owners of entities, and TPM manufacturer or designee*. |
| FMT_SMR.2.2 | The TSF shall be able to associate users with roles. |
| FMT_SMR.2.3 | The TSF shall ensure that the condition: *successful presentation of correct authentication data* is satisfied. |
| Dependencies: | FIA_UID.1 Timing of identification. |

### 5.2.6 Class FPT – Protection of the TOE Security Functions

#### FPT_AMT.1 Abstract machine testing

| | |
|---|---|
| Hierarchical to: | No other components |
| FPT_AMT.1.1 | The TSF shall run a suite of tests *during initial start-up and at the request of an authorized user* to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF. |
| Dependencies: | None. |
| *Application note:* | *The term "authorized user" in FPT_AMT.1 should be interpreted as any user. Authentication is NOT required for a user to run tests.* |

#### FPT_FLS.1 Failure with preservation of secure state

| | |
|---|---|
| Hierarchical to: | No other components |
| FPT_FLS.1.1 | The TSF shall preserve a secure state when the following types of failures occur: *failure of any crypto operations including RSA encryption, RSA decryption, SHA, RNG, RSA signature generation, HMAC generation; failure of any commands or internal operations*. |
| Dependencies: | ADV_SPM.1 Informal TOE security policy model |

#### FPT_RCV.4 Function recovery

| | |
|---|---|
| Hierarchical to: | No other components |
| FPT_RCV.4.1 | The TSF shall ensure that *all TPM Commands* have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state. |
| Dependencies: | ADV_SPM.1 Informal TOE security policy model |

#### FPT_RPL.1 Replay detection

| | |
|---|---|
| Hierarchical to: | No other components. |

| | |
|---|---|
| FPT_RPL.1.1 | The TSF shall detect replay for the following entities: *command requests that include the nonce parameter.* |
| FPT_RPL.1.2 | The TSF shall perform *destroy session* when replay is detected. |
| Dependencies: | None. |

**FPT_RVM.1** Non-bypassability of the TSP

| | |
|---|---|
| Hierarchical to: | No other components |
| FPT_RVM.1.1 | The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed. |
| Dependencies: | None. |

**FPT_SEP.1 TSF domain separation**

| | |
|---|---|
| Hierarchical to: | No other components |
| FPT_SEP.1.1 | The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects. |
| FPT_SEP.1.2 | The TSF shall enforce separation between the security domains of subjects in the TSC. |
| Dependencies: | None. |

**FPT_TDC.1 Inter-TSF basic TSF data consistency**

| | |
|---|---|
| Hierarchical to: | No other components |
| FPT_TDC.1.1 | The TSF shall provide the capability to consistently interpret *TPM Commands and responses* when shared between the TSF and another trusted IT product. |
| FPT_TDC.1.2 | The TSF shall use *the TCG Main Specification* when interpreting the TSF data from another trusted IT product. |
| Dependencies: | None. |

**FPT_TPMTST.1 TPM integrity test**

Hierarchical to: No other components.

| | |
|---|---|
| FPT_TPMTST.1.1 | The TSF shall run a suite of self-tests during initial start-up and at the request of an authorized user to demonstrate the correct operation of the TSF. |
| Dependencies: | None |

### 5.2.7 Class FTP – Trusted Path/Channels

**FTP_TRP.1 Trusted path**

| | |
|---|---|
| Hierarchical to: | No other components |
| FTP_TRP.1.1 | The TSF shall provide a communication path between itself and *local or remote* users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure. |
| FTP_TRP.1.2 | The TSF shall permit *the TSF, local or remote users* to initiate communication via the trusted path. |
| FTP_TRP.1.3 | The TSF shall require the use of the trusted path for *initial user authentication, for all TPM commands, all user commands, and TSF responses.* |
| Dependencies: | None |

### 5.2.8 Strength of Function Requirement

The threat level for the TOE authentication function is assumed to be SOF-Medium. The strength of cryptographic algorithms is outside the scope of the CC. Strength of function applies only to non-cryptographic, probabilistic or permutational mechanisms. The SOF requirement applies to the identification and authentication functionality within the TOE.

## 5.3 TOE Security Assurance Requirements

The Security Assurance Requirements for the TOE are the assurance components of Evaluation Assurance Level 3 (EAL3) augmented by ADV_SPM.1 and ALC_FLR.1. They are all drawn from Part 3 of the Common Criteria. The assurance components are listed in Table 6. EAL3 was selected because the TOE requires a moderate level of independently assured security and requires a thorough investigation of the TOE and its development without substantial re-engineering. ADV_SPM.1 was added because it is a dependency of functional security requirements FMT_MSA.2, FPT_FLS.1, and FPT_RCV.4. ALC_FLR.1 was added to provide basic flaw remediation.

*Table 6. EAL3 Assurance Requirements, augmented*

| | |
|---|---|
| ACM_CAP.3 | Authorization controls |
| ACM_SCP.1 | TOE CM coverage |
| ADO_DEL.1 | Delivery procedures |
| ADO_IGS.1 | Installation, generation, and start-up procedures |
| ADV_FSP.1 | Informal functional specification |
| ADV_HLD.2 | Security enforcing high-level design |
| ADV_RCR.1 | Informal correspondence demonstration |
| ADV_SPM.1 | Informal TOE security policy model [AUG] |
| AGD_ADM.1 | Administrator guidance |
| AGD_USR.1 | User guidance |

| | |
|---|---|
| ALC_DVS.1 | Identification of security measures |
| ALC_FLR.1 | Basic flaw remediation [AUG] |
| ATE_COV.2 | Analysis of coverage |
| ATE_DPT.1 | Testing: high-level design |
| ATE_FUN.1 | Functional testing |
| ATE_IND.2 | Independent testing - sample |
| AVA_MSU.1 | Examination of guidance |
| AVA_SOF.1 | Strength of TOE security function evaluation |
| AVA_VLA.1 | Developer vulnerability analysis |

# 6 TOE Summary Specification

## 6.1 TOE IT Security Functions

This section defines how the TOE satisfies the functional requirements defined in Section 5. The TOE provides the following functions, which are mapped to functional requirements classes and specific requirements in Table 7, below:

*Table 7. Security Functions and the SFRs They Implement*

| Security Function | Security Functional Requirements |
|---|---|
| Enforced Proof of Origin | FCP_NRO.2 |
| Cryptographic Support | FCS_CKM.1, FCS_CKM.4, and FCS_COP.1 (all iterations) |
| Manage User Access and Data | FDP_ACC.1, FDP_ACF.1, FDP_ETC.2, FDP_ITC.2, FDP_RIP.2 |
| Identification & Authentication | FIA_ATD.1, FIA_UAU.1, FIA_UAU.4, FIA_UAU.6, FIA_UID.1 |
| Security Management | FMT_MOF.1, FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, FMT_MTD.1, FMT_SMR.2 |
| Security Function Protection | FPT_AMT.1, FPT_FLS.1, FPT_RCV.4, FPT_RPL.1, FPT_RVM.1, FPT_SEP.1, FPT_TDC.1, FPT_TPMTST.1 |
| Trusted Path | FTP_TRP.1 |

The functionality is described in the subsections below, including rationale that the security functions are suitable to meet the Security Functional Requirements

### 6.1.1 Enforced Proof of Origin

The TOE meets the FCO_NRO.2, Enforced Proof of Origin requirement with commands that enforce generation of evidence. The AT97SC3201 commands that execute a sign operation and transmit signed data are: TPM_Sign, TPM_GetAuditEventSigned and TPM_GetCapabilitySigned; these commands enforce generation of evidence of origin by using the following procedure:

1. A valid user identity key must be loaded into the TPM using a TPM_LoadKey command. An authorization session must be opened using an OIAP or OSAP command. The LoadKey command requires demonstration of knowledge of the authorization secret for the parent key of that user key. Execution of any of the commands using the sign operation requires demonstration of knowledge of the authorization secret for that user key.

2. The authorization secret of the user identity key is used as an HMAC key during creation of an authorization digest for the returned digital signature. This HMAC operation is performed on a combination of the following parameters:

    a. A SHA1 hash of the following values:

        i. The command code for the TCG operation

    ii. The length of the returned digital signature

    iii. The digital signature

b. An “Even” nonce newly generated by the TPM

c. The “Odd” nonce generated by the system and transmitted to the TPM with the command ordinal.

d. The continueAuthSession flag, indicating whether the authorization session should remain active after execution of the present command.

The system receiving the digital signature recreates the HMAC operation on the same data and compares the results. If the results are identical, the system is assured that the signature was generated by an entity that had knowledge of the authorization information for the user identity key. Only the TPM and the user authorizing the digital signature operation share this authorization information.

The TSF verifies that the originator of the data to be signed has knowledge of the user key authorization information. Since that authorization information is used as the key for an HMAC operation performed on data that includes the data to be signed, the identity of the originator is tied to the data itself.

The signature returned by the TSF is accompanied by an authorization digest that is created by an HMAC operation that includes as input parameters nonces that have been generated by the originator and by the TSF during each step of the signature command sequence. These nonces are available only after a proper authorization session (either OIAP or OSAP) has been opened by the requestor.

### 6.1.2 Cryptographic Support

The TOE provides cryptographic support that meets the requirements for cryptographic key generation, cryptographic key destruction, and cryptographic operation. The TOE functions are described below with a reference to each CC functional requirement claimed for cryptographic support.

- FCS_CKM.1 Cryptographic Key Generation: The TOE generates cryptographic keys in accordance with the cryptographic key generation algorithm RSA, and cryptographic key sizes RSA 512, 1024, 2048 that meet the following: PKCS#1 V2.
- FCS_CKM.4 Cryptographic Key Destruction: The TOE disables access to cryptographic keys in accordance with a specified cryptographic key destruction method. Register bits in TCPA_KEY_FLAGS that indicate the valid/invalid status of cryptographic keys are reset. Nonvolatile memory locations containing the actual key data are marked with a tag indicating “free memory space”.
- FCS_COP.1:1 Cryptographic Operations; RSA encrypt and decrypt: The TOE performs encryption and decryption in accordance with a specified cryptographic algorithm RSA and cryptographic key sizes RSA 512, 1024, 2048 that meet the following: PKCS#1 V2.
- FCS_COP.1:2 Cryptographic operation; RSA signature and signature verification: The TOE performs signature generation and signature verification in accordance with the cryptographic algorithm RSA and cryptographic key sizes RSA 512, 1024, 2048 that meet the following: PKCS#1 V2.
- FCS_COP.1:3 Cryptographic operation; SHA: The TOE performs secure hash in accordance with the cryptographic algorithm SHA-1 that meet the following: FIPS 180-1.

- FCS_COP.1:4 Cryptographic operation; Keyed-Hashing for Message Authentication: The TOE performs keyed-hashing message authentication code (HMAC) in accordance with a specified cryptographic algorithm SHA-1 and cryptographic key sizes 160 bits that meet the following: RFC 2104.
- FCS_COP.1:5 Cryptographic operation – RNG: The TOE performs random number generation in accordance with a specified cryptographic algorithm random number generator and cryptographic key sizes not applicable that meet the following: TCG Main Specification version 1.1b, Section 10.5.

### 6.1.3 Manage User Access and Data

The TOE provides access control by enforcing the Protected Operations Access Controls on Subjects (commands executing on behalf of users), Objects (keys and user data), and Operations (signature generation, encryption, or decryption) by requiring authorization before execution of commands involving protected operations. This authorization is given by the TSF only after the requestor has demonstrated knowledge of the appropriate user authorization secret information during execution of the LoadKey command, and again when any command that uses the loaded key is transmitted to the TPM. This functionality meets FDP_ACC.1.

The TOE meets FDP_ACF.1 by enforcing the Protected Operations Access Controls on objects based on security attributes stored as nonvolatile fields within a keyInfo structure that is created when the key is originally generated by the TSF. The attributes recorded in the keyInfo structure include:

1. TCPA_AUTH_DATA_USAGE – This flag may be set to TPM_AUTH_NEVER if the key does not require authorization. If the flag is set to TPM_AUTH_ALWAYS, then the required authorization information must be provided with the input parameters as specified in the command definition. The required authorization parameters are specified with the input parameter list designated for each command in the TCPA main specification, version 1.1b. No values are accepted by the TSF except TPM_AUTH_ALWAYS (0x01) and TPM_AUTH_NEVER (0x00).

2. TCPA_KEY_FLAGS – This data structure defines the capabilities of keys used in the commands TCPA_STORE_ASYMKEY and TCPA_CERTIFY_INFO. A mask value of 0x00000001 indicates the use of redirected output. A mask value of 0x00000002 indicates that the key is migratable. A mask value of 0x00000004 indicates that the key is volatile, and must be unloaded upon execution of the TPM_Init/TPM_Startup sequence. No values are accepted by the TSF except these three values (redirection, migration, volatile).

3. TCPA_KEY_USAGE – This structure defines the key type, which determines the choices of encryption and signature schemes allowable for that key. The list of acceptable key types include:

   a. TPM_KEY_SIGNING (0x010) – A signing key. The private portion of this key pair is used for signing operations only. The key must be stored as a leaf (not a parent key) in the TSF Protected Storage hierarchy.

   b. TPM_KEY_STORAGE (0x011) – A storage key. The key is used only to wrap and unwrap other keys in the Protected Storage hierarchy.

   c. TPM_KEY_IDENTITY (0x012) – An Identity key. The key is used only for operations that require a TPM identity.

   d. TPM_KEY_AUTHCHANGE (0x013) – An ephemeral key that is used only during the ChangeAuthAsym process.

**e.** TPM_KEY_BIND (0x014) – A binding key. The key can be used only for TPM_Bind and TPM_Unbind operations.

**f.** TPM_KEY_LEGACY (0x015) – Legacy keys are allowed to perform both signing and binding operations. It is a deprecated capability provided to support applications developed for earlier versions of the TPM which allowed both signing and encryption operations to be performed by the same key.

The TSF enforces the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- Key and data access is defined as "owner" access or "world" based on the value of TCPA_AUTH_DATA_USAGE. If the value indicates that authorization is not required, access is not restricted and is available to the world. If the value indicates that owner authorization is required, then access is restricted to the owner of the authorization secret. If unauthorized access is requested for a key or data that requires authorization, the TSF will return the error code: TCPA_AUTHFAIL.

- Cryptographic operations for each key are limited based on the specification of the TCPA_KEY_USAGE value. For each TSF operation, the TPM determines the value of the TCPA_KEY_USAGE parameter for any keys used by that operation. The key type determines whether or not a key can be used for the specified operation. If the operation is not allowed for the specified key type, the TSF will return the error code: TCPA_INVALID_KEYUSAGE.

The TOE protects export of data and meets FDP_ETC.2.1 by the following means:

- The TSF enforces the Protected Operations Access Controls when exporting user data, controlled under the SFP, outside of the TSC by requiring knowledge of the user authorization secret for access to any data or keys.
- The TSF exports the user data with the user data's associated security attributes. Keys are exported in a data structure designated TCPA_KEY, which contains all the key security attributes. The component parameters of TCPA_KEY are described in the table below:

| **Type** | **Name** | **Description** |
|---|---|---|
| `TCPA_VERSION` | `ver` | Version number |
| `TCPA_KEY_USAGE` | `keyUsage` | TCPA key usage that determines the operations permitted with this key |
| `TCPA_KEY_FLAGS` | `keyFlags` | Indication of migration, redirection etc. |
| `TCPA_AUTH_DATA_USAGE` | `authDataUsage` | Indicates the conditions where it is required that authorization be presented. |
| `TCPA_KEY_PARMS` | `algorithmParms` | Information regarding the algorithm for this key |
| `UINT32` | `PCRInfoSize` | Length of the pcrInfo parameter. If the key is not bound to a PCR this value is 0. |
| `BYTE*` | `PCRInfo` | Structure of type TCPA_PCR_INFO, or an empty array if the key is not bound to PCRs. |

| | | |
|---|---|---|
| `TCPA_STORE_PUBKEY` | `pubKey` | Public portion of the key. |
| `UINT32` | `encSize` | Size of the encData parameter. |
| `BYTE*` | `encData` | An encrypted TCPA_STORE_ASYMKEY structure or TCPA_MIGRATE_ASYMKEY structure |

- The TSF ensures that the security attributes, when exported outside the TSC, are unambiguously associated with the exported user data. Data can only be exported as a direct result of successful execution of a single TPM command that also contains complete authorization information for access to the particular data or key. During execution of the command, the TSF evaluates the authorization information and, if authorization is successful, calculates an HMAC value that accompanies the exported data. This value guarantees that the security attributes of the exported data have been correctly taken into account during command execution, and have been used to determine that the target data has the proper security attributes to allow exportation. The HMAC is calculated from the following information:

  1. The return code of the operation
  2. The command ordinal
  3. The size of the exported data blob
  4. The exported data blob itself
  5. A random authorization nonce generated by the TPM, used for anti-replay protection
  6. The authorization nonce generated by the system that provided the authorization for the key or data, which was transmitted to the TSF with the command
  7. A Boolean value indicating that the authorization session will (1) or will not (0) continue after completion of the current command
  8. A random data-nonce generated by the TPM associated with the data to be exported
  9. A data-nonce generated by the system that provided the authorization for the key or data, which was transmitted to the TSF with the command
  10. A Boolean value indicating that the data-transmission session will (1) or will not (0) continue after completion of the current command

- The TSF enforces the following rules when user data is exported from the TSC: *A key may be encrypted for migration only if the migratable flag is set in TCPA_KEY_FLAGS.* When a key is generated (TPM_CreateWrapKey), the migratable flag in TCPA_KEY_FLAGS is set to 0 if the key is designated as non-migratable. A key that is designated as non-migratable is protected against migration by installing an unknowable secret (tpmProof) as the migration authorization value in the TCPA_STORE_ASYMKEY structure.

The TSF enforces the Protected Operations Access Controls when importing user data, thus meeting FDP_ITC.2, controlled under the SFP, from outside of the TSC. Authorization verification must be successfully completed by the TSF before any action is taken on imported user data. All TSF command sequences are protected against replay attacks by using a conventional method of generation/verification of random nonce values that accompany every communication block between the TSF and the outside world.

The TSF uses the security attributes associated with the imported user data. The security attributes associated with specific user data are directly tied to the user authorization procedure that must be completed successfully by the TSF before any action can be taken on the data.

The TSF ensures that the protocol used provides for the unambiguous association between the security attributes and the user data received. The same authorization protocol is used for all TSF communications. The communication block that contains user data must be preceded by the authorization information for that data. Actions taken on user data are allowed only after completion of successful authorization for that data. The security attributes for the authorized data are stored with the user data in the same data structure and are verified during command execution.

The TSF ensures that interpretation of the security attributes of the imported user data is as intended by the source of the user data. The same authorization protocol is used for all TSF communications. The communication block that contains user data must be preceded by the authorization information for that data. Actions taken on user data are allowed only after completion of successful authorization for that data. The security attributes for the authorized data are stored with the user data in the same data structure and are verified during command execution, which assures that all actions taken on the data use the stored security attributes as intended. Security attributes are imported with data as part of the encrypted blob. As part of the blob that has been encrypted, the security attributes are unambiguously associated with the data.

Full residual information protection, meeting FDP_RIP.2, is provided as follows: the TSF ensures that any previous information content of a resource is made unavailable upon the de-allocation of the resource from all objects. Availability of key data is permanently removed when keys are unloaded or deleted. Register bits in TCPA_KEY_FLAGS that indicate the valid/invalid status of cryptographic keys are reset. Nonvolatile memory locations containing the actual key data are marked with a tag indicating "free memory space".

### 6.1.4 Identification And Authentication

The TPM identification and authentication capability is used to authenticate an entity owner and to authorize use of an entity. Note that the TCG Main Specification document refers to the identification and authentication process and this data as "authorization". The basic premise is to prove knowledge of a shared secret when a command requiring authorization is passed to the TPM. The authorization data is a shared secret between the TPM and the owner of the entity. The data is an 8-byte password. Authorization data is created and associated with the TPM Owner and each entity (keys, for example) that the TPM controls. The authorization data for the TPM Owner and the Storage Root Key are held within the TPM itself and the authorization data for other entities are held with the entity.

There is a separate password (authorization data) for each entity. The TPM Owner authorization data, which is defined as part of taking ownership of the TPM, allows the Owner to prove ownership of the TPM and to perform certain commands that are available only to the TPM Owner. Proving ownership of the TPM does not allow access to all entities – the TPM Owner is not a "super user" and additional authorization data must be provided for each entity or operation that has protection.
The TPM treats knowledge of the authorization data as complete proof of ownership of the entity.

To meet FIA_ATD.1, the TSF maintains the following list of security attributes belonging to individual users: authentication data, role. The authentication data is maintained in a data structure TCPA_STORE_ASYMKEY that is stored with the key when it is generated. Authentication data is known as: usageAuth. Authentication data is passed to the TOE by the user with each command that requires authentication data. The TOE allows access to data and keys where the entity owner has given the "world" access based on the value of TCPA_AUTH_DATA_USAGE; and access to the following commands: *TPM_SelfTestFull, TPM_ContinueSelfTest, TPM_GetTestResult, TPM_PcrRead, TPM_DirRead, TPM_EvictKey, TPM_DisableForceClear, TPM_CreateEndorsementKeyPair, TPM_Extend, TPM_GetCapability, TPM_GetOrdinalAuditStatus, TPM_OIAP, TPM_OSAP, TPM_ReadPubek, TPM_Reset, TPM_SaveState, TPM_SetOwnerInstall, TPM_GetRandom, TPM_SetTempDeactivated, TPM_SHA1Complete, TPM_SHA1CompleteExtend, TPM_SHA1Start, TPM_SHA1Update, TPM_Startup, TPMStirRandom, TPM_TerminateHandle*, TPM_SetState, TPM_GetState, TPM_Identify, TPM_VerifySignature, and TPM_BindV20 on behalf of the user to be performed before the user is authenticated or identified. Each user must be successfully authenticated and identified before allowing any other TSF-mediated actions on behalf of that user. If TSF-mediated actions are requested without successfully completing the required authentication and identification (.known as authorization within the TCG specification), the TSF will return the error code: TCPA_AUTHFAIL. This functionality meets FIA_UAU.1 and FIA_UID.1 requirements. The TOE re-authenticates a user for every command that requires user authentication as specified in the TCG specification, thereby meeting the FIA_UAU.6 requirement.

The TOE provides a single-use authentication mechanism that prevents reuse of authentication data related to the use of the "Object-Independent Authorization Protocol" (OI-AP) and the "Object-Specific Authorization Protocol" (OS-AP) protocols. When use of an authentication handle by a valid command is concluded, access to that handle is denied until a new OIAP or OSAP session is established. The use of random nonces, which accompany the commands that establish new OIAP or OSAP sessions, prevent reuse or replay of previous data. This functionality meets FIA_UAU.4.

In an effort to stave off authorization-based attacks, the TPM accumulates the total number of failed authorization attempts on any entity in the FailCount register. Statistically, some failed authorization attempts will occur under normal usage and because of this, the failure counter mechanism involves two stages. Failed attempts up to the value of the failure modulus do not cause any lockout.

The very next failure, however, causes a delay in the form of a lockout period. After the delay times out, another <failure modulus> attempts are permitted before the next delay is imposed. The length of the delay increases geometrically each time with the first delay lasting 1.1 minutes, the second lasting 2.2 minutes, and so on. The value of the failure modulus is controlled by the entry in the FAILMOD register, according to the table below. Values greater than 10 are not permitted by the chip. The evaluated configuration of the chip includes a FAILMOD set to 1.

| 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 following table shows the number of failures that would have to occur to cause the chip to lock up for the specified period of time.

## Example of FAILCOUNT values

| FAILCOUNT ÷ *2* $^{\text{FAILMOD}}$ | FAILMOD Value | | | | | | | | | | **Approx Delay on Next Failure** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | |
| **0x00** | 1 | 3 | 7 | 15 | 31 | 63 | 127 | 255 | 511 | 1023 | **1.1 minutes** |
| **0x01** | 3 | 7 | 15 | 31 | 63 | 127 | 255 | 511 | 1023 | 2047 | **2.2 minutes** |
| **0x02** | 5 | 11 | 23 | 47 | 95 | 191 | 383 | 767 | 1535 | 3071 | **4.4 minutes** |
| **0x03** | 7 | 15 | 31 | 63 | 127 | 255 | 511 | 1023 | 2047 | 4095 | **8.8 minutes** |
| **0x04** | 9 | 19 | 39 | 79 | 159 | 319 | 639 | 1279 | 2559 | 5119 | **17.6 minutes** |
| **0x05** | 11 | 23 | 47 | 95 | 191 | 383 | 767 | 1535 | 3071 | 6143 | **35.2 minutes** |
| **0x06** | 13 | 27 | 55 | 111 | 223 | 447 | 895 | 1791 | 3583 | 7167 | **1.2 hours** |
| **0x07** | 15 | 31 | 63 | 127 | 255 | 511 | 1023 | 2047 | 4095 | 8191 | **2.3 hours** |
| **>= 0x08** | 17 | 35 | 71 | 143 | 287 | 575 | 1151 | 2303 | 4607 | 9215 | **4.7 hours** |

Entries in this table represent the number of failed authorization attempts at a given FAILMOD value that will cause a specific delay. For example, if FAILMOD is set to 5 and FAILCOUNT is 159 (signifying that 159 failed authorization attempts have occurred), then on the next failure (the 160th) the TPM will lock up for 17.6 minutes. If the next attempt (the 161st) fails, there will be no lockout delay. The next lockout delay will occur after the 191st failure.

The chip does not increase the lockout interval beyond 4.7 hours. For every <failure modulus> failures beyond this point, the chip still locks up for another 4.7 hours. Note that, while in lockout mode, the chip does NOT permit the system clock to be halted by using CLKRUN#. If the user goes into sleep state while the lockout counter is active, the count will start all over again when the user wakes the TPM up.

### 6.1.5 Security Management

The TOE manages security functions behavior for specific commands, thereby meeting FMT_MOF.1. The TOE restricts the ability to *disable or enable* the functions to:

- *Reset the Authorization Failure Counter to 0x0000 (TPM_FAILCOUNT).* TPM_OwnerSetState is an Atmel-specific command that may be used to reset the Authorization Failure Attempt Counter to 0x0000 once per delay session as described in 6.1.4 above. TPM Owner authorization is required to execute this command.
- *Change the operating mode of the Authorization Failure Counter (TPM_FAILMOD).* TPM_OwnerSetState is an Atmel-specific command that may be used to set the operating mode (mod) of the Authorization Failure Attempt Counter. TPM Owner authorization is required to execute this command.
- *Initialize and lock the FIPS operating mode (TPM_FIPS)* to the TPM owner. TPM_OwnerSetState is an Atmel-specific command that may be used to set and lock the FIPS-mode bit. Owner authorization is required to execute this command.

Security attributes are managed by the TOE through the Protected Operations Access Controls, which restrict the ability to create the security attributes associated with a particular entity, including TCPA_KEY_USAGE, TCPA_AUTH_DATA_USAGE, migratable flag, and volatility flag to the entity owner. These security attributes are specified by input parameters to the TPM_CreateWrapKey command. Authorization data for the parent key is required to execute this command. This functionality meets the requirement FMT_MSA.1.

The TOE meets FMT_MSA.2, Secure security attributes, by ensuring that only secure values are accepted for security attributes. Security attributes are specified by input parameters to the TPM_CreateWrapKey command. Owner authorization is required to execute this command. Security attributes are thereby assigned as directed by the owner, and apply only to the target key created during execution of the command. It is not possible for attributes of one key to affect the security parameters of another key or compromise the security of any other user data or keys.

Static attribute initialization, which meets FMT_MSA.3, is enforced by the Protected Operations Access Controls, which provide specific default values for security attributes that are used to enforce the SFP. Default values are defined as follows for the TPM_MakeIdentity command.

- TCPA_KEY_USAGE
    - keyUsage = TPM_KEY_IDENTITY
    - algorithm type = RSA
    - key length = 2048 bits
- TCPA_KEY_FLAGS
    - Migratable flag = FALSE

The TSF allows the *entity owner* to specify alternative initial values to override the default values when an object or information is created. With the exception of the default security parameters listed in FMT_MSA.3.1 for TPM_MakeIdentity, the entity owner explicitly specifies all security attributes when TPM_CreateWrapKey creates a new key. These attributes are specified as TCPA_KEY parameters passed to the TSF with the TPM_CreateWrapKey command.

The management of TSF data, meeting iterations of FMT_MTD.1, is performed according to the role and the function performed as follows:

For the role TPM Owner and the function modify the TOE restricts the ability to modify the TSF data: Identification and authentication data associated with the Endorsement Key and SRK; Migration authorization data to the TPM Owner. The Endorsement Key is generated prior to establishment of an Owner, and is used in the process of TPM_TakeOwnership. Once created, there is no means to modify or delete the Endorsement Key by the Owner or by any other entity or identity. The Storage Root Key authorization data and/or the Owner authorization data can be modified by execution of the TPM_ChangeAuthOwner command. Presentation of the current Owner authorization data is required to authorize execution of the TPM_ChangeAuthOwner command.

For the role TPM Owner and the function create, the TOE restricts the ability to generate the TSF data: Storage Root Key and TPMProof to the TPM Owner. Both the Storage Root Key and TpmProof are generated by the TSF during execution of the TPM_TakeOwnership command. Presentation of the Owner authorization data is required to authorize execution of the TPM_TakeOwnership command.

For the role Entity Owner, the TOE restricts the ability to *modify* the *TSF data: Identification and Authentication data associated with entity;* to the *entity Owner*. Modification of authentication data associated with an entity is performed by execution of the TPM_ChangeAuth command. Presentation of entity authorization data for both the key to be modified and the parent of that key is required to authorize execution of the TPM_ChangeAuth command.

For the role Manufacturer, the TOE restricts the ability to *generate* the *TSF* data: *Endorsement Key Pair* to the *TPM manufacturer or designee*. The TSF is shipped with no Endorsement Key Pair generated or present on board the TSF. Atmel designates the platform manufacturer as the authorized entity to execute the TPM_CreateEndorsementKeyPair command after assembly of the platform is completed.

The TOE meets FMT_SMR.2 by maintaining the following roles: TPM owner, owners of entities, and TPM manufacturer or designee. The role of TPM owner is defined as the entity that knows and can successfully present the owner authorization data. The role of entity owner is defined as the entity that knows and can successfully present the entity authorization data. The TPM manufacturer for the AT97SC3201 is Atmel Corporation, which is encoded into the TSF register TCPA_CAP_PROP_MANUFACTURER as the ASCII value: ATML. Atmel assigns no designee.

Users are associated with roles. The role of TPM owner is defined as the entity that knows and can successfully present the owner authorization data. The role of entity owner is defined as the entity that knows and can successfully present the entity authorization data.

The TOE verifies successful presentation of correct authentication data for every command that requires authentication. Commands for the TSF fall into three categories:

1. Commands that require authorization of one entity.
2. Commands that require authorization of two entities.
3. Commands that do not require authorization (this section of the ST does not apply to these commands).

There are two authorization protocols used by the TSF: the “Object Independent Authorization Protocol” (OIAP) and the “Object Specific Authorization Protocol” (OSAP). Presentation of authentication data for OIAP sessions uses the following procedure:

- Open an OIAP session by executing TPM_OIAP. The TSF generates and returns two values: authHandle and nonceEven.
- Authorization information for a specific target entity is then passed into the TSF as parameters of the subsequent command. These parameters include:
    1. keyHandle – the handle, assigned by the TPM, that refers to a key associated with the target entity that was previously loaded by the command TPM_LoadKey.
    2. authHandle – the handle assigned by the TPM for the current OIAP session.
    3. authLastNonceEven – the nonce generated by the TPM on the previous command to cover inputs to the TPM on the subsequent command.
    4. nonceOdd – the nonce generated by the system associated with authHandle.
    5. continueAuthSession – a Boolean value that indicates whether the OIAP session will continue for the subsequent command.
    6. The authorization digest of type TCPA_AUTHDATA, calculated by the system for the target entity, using the inputs and keyHandle in this OIAP session.
- If two authorization sessions are required for the command being executed, the following additional parameters will be passed to the TSF to provide authorization information for the second session:
    1. keyHandle – the handle, assigned by the TPM, that refers to a key associated with the second target entity that was previously loaded by the command TPM_LoadKey.

2. entityLastNonceEven – the nonce generated and returned by the TPM when the authorization session was opened for the second entity.
3. entityNonceOdd – the nonce generated by the system when the authorization session was opened for the second entity.
4. entityContinueAuthSession – a Boolean value that indicates whether the authorization session for the second entity will continue for the subsequent command..
5. The authorization digest of type TCPA_AUTHDATA, calculated by the system for the second target entity, using the inputs and keyHandle in this OIAP session.

- The TSF forms the authorization digest by performing the HMAC calculation for Authorization
  1. AuthDigest1 = HMAC( <paramDigest>, EvenNonce1, OddNonce1, continueUse1)
  2. AuthDigest2 = HMAC( <paramDigest>, EvenNonce2, OddNonce2, continueUse2)

  <paramDigest> is the output result of a SHA-1 hash of all authorized parameters other than the authorization setup parameters (authHandle, nonces and continueAuthSession). If two authorization sessions are required for a particular command, <paramDigest> is identical for both sessions.
- The TSF compares the authorization digest calculated internally with the digest passed by the system as a command parameter. If the digests are equal, authorization is successful. If the digests are not equal, authorization has failed and the TSF will return the error code: TCPA_AUTHFAIL. If two authorization sessions are required, both digests are compared independently and both authorization sessions must pass. If the second authorization fails, the TSF will return the error code: TCPA_AUTH2FAIL.

The second authorization protocol is the Object-Specific Authorization Protocol (OSAP). OSAP creates an ephemeral secret that is used in place of the entity authorization secret throughout the OSAP session. OSAP sessions may be extended to multiple commands that operate on a single authorized entity by use of the continueAuthSession parameter passed with the command by the system to the TSF. Presentation of authentication data for OSAP sessions uses the following procedure:

- Open an OSAP session by executing TPM_OSAP. TPM_OSAP must contain the following input parameters:
  1. entityType – the target key type (either keyHandle from a previously loaded key, the Owner authorization data or the Storage Root Key authorization data).
  2. entityValue – the handle number of a previously loaded key (assigned by the TSF when the key is loaded). If the entityType is either the Owner data or SRK data, then entityValue is ignored.
  3. nonceOddOSAP – a nonce generated by the caller associated with the shared secret.
- The TSF generates and returns three values: authHandle, nonceEven and nonceEvenOSAP.
- Authorization information for a specific target entity is then passed into the TSF as parameters of the subsequent command. These parameters include:
  1. keyHandle – the handle, assigned by the TPM, that refers to a key associated with the target entity that was previously loaded by the command TPM_LoadKey.
  2. authHandle – the handle assigned by the TPM for the current OSAP session.
  3. authLastNonceEven – the nonce generated by the TPM on the previous command to cover inputs to the TPM on the subsequent command.
  4. nonceOdd – the nonce generated by the system associated with authHandle.

    5. continueAuthSession – a Boolean value that indicates whether the OSAP session will continue for the subsequent command.

    6. The authorization digest of type TCPA_AUTHDATA, calculated by the system for the target entity, using the inputs and keyHandle in this OSAP session.

- The TSF calculates the shared secret using an HMAC calculation. The key for the HMAC calculation is the secret authorization data assigned to the key handle identified by the input command parameter entityValue.
- The TSF forms the authorization digest by performing the HMAC calculation for Authorization
    1. AuthDigest1 = HMAC( <paramDigest>, EvenNonce1, OddNonce1, continueUse1)
    2. AuthDigest2 = HMAC( <paramDigest>, EvenNonce2, OddNonce2, continueUse2)

    <paramDigest> is the output result of a SHA-1 hash of all authorized parameters other than the authorization setup parameters (authHandle, nonces and continueAuthSession). If two authorization sessions are required for a particular command, <paramDigest> is identical for both sessions.
- The TSF compares the authorization digest calculated internally with the digest passed by the system as a command parameter. If the digests are equal, authorization is successful. If the digests are not equal, authorization has failed and the TSF will return the error code: TCPA_AUTHFAIL. If two authorization sessions are required, both digests are compared independently and both authorization sessions must pass. If the second authorization fails, the TSF will return the error code: TCPA_AUTH2FAIL.

### 6.1.6 Security Function Protection

Security function protection includes abstract machine testing that meets the FPT_AMT.1 requirement. The TSF runs a suite of tests during initial start-up and at the request of an authorized user to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF. This meets the FPT_TPMTST.1 requirement. The term "authorized user" in FPT_AMT.1 and FPT_TPMTST.1 should be interpreted as any user. Authentication is NOT required for a user to run tests. On initial powerup, the TSF performs a self-test of the SHA-1 algorithm only. Control of the TSF is then returned to the system. However execution of any commands is restricted until execution of the command TPM_ContinueSelfTest has successfully completed the remainder of the self-test suite. When any command is issued that calls a capability or function that has not yet been tested, the TSF will ignore the input command and automatically execute the remainder of the self-test suite instead. In this case, the TSF will not respond to the caller with a return code. Commands issued in all other circumstances will result in a return code from the TSF. The remaining suite of self-tests may be explicitly executed by running the command TPM_ContinueSelfTest.

The full test suite (initial POST plus ContinueSelfTest) consists of tests of the following TSF functions:

1. SHA-1 hash function (POST)
2. RSA encryption/decryption (ContinueSelfTest)
3. RSA Signature/verify operations (ContinueSelfTest)
4. Random number generator (ContinueSelfTest)
5. Key generation (ContinueSelfTest)

The TSF preserves a secure state when the following types of failures occur, thereby meeting FPT_FLS.1: failure of any crypto operations including RSA encryption, RSA decryption, SHA, RNG, RSA signature generation, HMAC generation; failure of any commands or internal operations. In the event of a failure of any operation, the TSF will return a failure code only (following the requisite tag that indicates a return operation and the total number of returned bytes). No additional information is returned by the TSF. Errors designated as fatal errors will terminate any active authorization sessions. Nonfatal errors do not terminate authorization sessions.

The TSF provide function recovery, meeting FPT_RCV.4, by the following means: all TPM Commands have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state. The TPM always returns to a standby state after completion of a command sequence. If the command results in an error, the TPM returns a response tag, the number of response bytes, and an error code. It then enters the standby state, awaiting the next command from the system. If the command execution is completed successfully, the TPM enters the standby state after returning the appropriate response. If unsuccessful, the TPM returns an error code and enters a standby state, waiting for a new command.

The TOE meets FPT_RPL.1. by detecting replay for the following entities: command requests that include the nonce parameter. The TPM protocols use a "rolling nonce" paradigm. This requires that a nonce from one side be in use only for a message and its reply. For instance, the TPM would create a nonce and send that on a reply. The requestor would receive that nonce and then include it in the next request. The TPM would validate that the correct nonce was in the request and then create a new nonce for the reply. This mechanism is in place to prevent replay attacks.

A session is destroyed when replay is detected. If a response or command is detected by the TSF that does not contain the appropriate nonces, the TSF will return the error code TCPA_BAD_PARAMETER or TCPA_AUTHFAIL. The current authorization session will be terminated and the TSF will enter a standby mode.

The TOE provides non-bypassability of the TSP, thereby meeting FPT_RVM.1, by ensuring that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed. Command execution does not begin on any TSF function until all required authorization sequences are completed successfully.

The TOE maintains domain separation, meeting FPT_SEP.1, by maintaining a security domain for its own execution that protects it from interference and tampering by untrusted subjects. Command execution does not begin on any TSF function until all required authorization sequences are completed successfully. Following execution of the command, either successful or unsuccessful, the TSF will terminate the current authorization session and enter a standby state, unless specifically directed by the owner of the entity (see continueAuth bit). The TSF enforces separation between the security domains of subjects in the TSC. Access is granted to an entity within the TSF only after successful completion of the authorization protocol for that target entity. Access to any data not associated with the current authorization sessions is not allowed. Following execution of the command, either successful or unsuccessful, the TSF will terminate the current authorization session and enter a standby state.

Inter-TSF data consistency is provided, meeting FPT_TDC.1, by the TOE providing the capability to consistently interpret TPM commands and responses when shared between the TSF and another trusted IT product. All TSF commands and responses are specified in the TCG Main Specification, version 1.1b, and the Atmel-Specific Commands document, version 0.17. No other commands exist, nor are any interpretations allowed other than those written in these specification documents. The TSF uses the TCG Main Specification when interpreting the TSF data from another trusted IT product. All TSF commands and responses are specified in the TCG Main Specification, version 1.1b, and the Atmel-Specific Commands document, version 0.17. No other commands exist, nor are any interpretations allowed other than those written in these specification documents.

### 6.1.7 Trusted Path

The TSF meets the FTP_TRP. 1 requirement by providing a communication path between itself and local or remote users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure. The TSF communicates to the system through the LPC bus interface. The LPC command protocol includes a device address with every command, which provides assurance that a command issued by the system is intended for the TSF. Data is communicated to the TSF only through TCG commands, which require successful presentation of authorization data before execution by the TSF.

Sensitive data communicated to the TSF is encrypted to protect it from modification or disclosure. Two protocols are utilized by the TSF for the purposes of creating the encrypted blobs. The ADIP procedure is used when communicating authorization information for new entities. The ACIP procedure is used when changing existing authorization information. The following descriptions of ADIP and ACIP are extracted from the TCG Main Specification version 1.1b:

**ADIP – Creating a New Entity**

The creation of the authorization data is the responsibility of the entity owner. He or she may use whatever process he or she wishes. The transmission of the authorization data from the owner to the TPM requires confidentiality and integrity. The encryption of the authorization data meets these requirements. The confidentiality and integrity requirements assume the insertion of the authorization data occurs over a network. While local insertions of the data would not require these measures, the protocol is established to be consistent with both local and remote insertions.

When the requestor is sending the authorization data to the TPM, the command to load the data requires the authorization of the entity owner. For example, to create a new TPM ID and set its authorization data requires the authorization data of the TPM Owner.

The confidentiality of the transmission comes from the encryption of the authorization data, and the integrity comes from the ability of the owner to verify that the authorization is being sent to a TPM and that only a specific TPM can decrypt the data.

The mechanism uses the following features of the TPM, OS-AP and HMAC.

The creation of a new entity requires the authorization of the entity owner. When the requestor starts the creation process, the creator must use OS-AP.

The creator builds an encryption key using a SHA-1 hash of the shared secret from the OS-AP mechanism and the nonce (authLastNonceEven) returned by the TPM from the TPM_OSAP command.

The creator encrypts the new authorization data using the key from the previous step as a one-time pad with XOR and then sends this encrypted data along with the creation request to the TPM.

The TPM decrypts the authorization data using the OS-AP shared secret and authLastNonceEven, creates the new entity.

The TPM sends the reply back to the creator using the new authorization data as the secret value of the HMAC.

The creator believes that the OS-AP creates a shared secret known only to the creator and the TPM. The TPM believes that the creator is the entity owner by their knowledge of the parent entity authorization data. The creator believes that the process completed correctly and that the authorization data is correct because the HMAC will only verify with the OS-AP secret.

The ADIP allows for the creation of new entities and the secure insertion of the new entity authorization data. The transmission of the new authorization data uses encryption with the key being a shared secret of an OS-AP session.

The OS-AP session must be created using the owner of the new entity.

In the following example, we want to send the previously described command TPM_EXAMPLE to create a new entity. In the example, we assume there is a third input parameter newAuth, and that one of the input parameters is named parentHandle to reference the parent for the new entity (TPM Owner in some circumstances such as the SRK and its children, otherwise a key).

| **Caller** | **On the wire** | **Dir** | **TPM** |
|---|---|---|---|
| Send TPM_OSAP | TPM_OSAP<br>parentHandle<br>nonceOddOSAP | → | Create session & authHangle<br>Generate authLastNonceEven<br>Save authLastnonceEven with authHandle<br>Generate nonceEvenOSAP<br>Generate sharedSecret = HMAC(parent.usageAuth, nonceEvenOSAP, nonceOddOSAP)<br>Save parentHandle, sharedSecret with authHandle |
| Save authHandle, authLastNonceEven<br>Generate sharedSecret = HMAC(parent.usageAuth, nonceEvenOSAP, nonceOddOSAP)<br>Save sharedSecret | authHandle, authLastNonceEven<br>nonceE venOSAP | ← | Returns |
| Generate nonceOdd & save with authHandle.<br>Compute input parameter newAuth = XOR( entityAuthData, SHA1(sharedSecret, authLastNonceEven))<br>Compute inAuth = HMAC (sharedSecret, inParamDigest, inAuthSetupParams) | | | |
| Send TPM_Example | tag<br>paramSize<br>ordinal<br>inArgOne<br>inArgTwo<br>newAuth<br>authHandle<br>nonceOdd<br>continueAuthSessio n<br>inAuth | → | Verify authHandle points to a valid session, mismatch returns TPM_AUTHFAIL<br>Retrieve authLastNonceEven from internal session storage<br>HM = HMAC (sharedSecret, inParamDigest, inAuthSetupParams)<br>Compare HM to inAuth. If they do not compare return with TPM_AUTHFAIL<br>Compute entityAuthData = XOR( newAuth, SHA1(sharedSecret, authLastNonceEven))<br>Execute TPM_Example, create entity and build returnCode<br>Generate nonceEven to replace authLastNonceEven in session |

| | | | |
|---|---|---|---|
| | | | Set resAuth = HMAC(sharedSecret, outParamDigest, outAuthSetupParams) |
| Save nonceEven<br>HM = HMAC( sharedSecret, outParamDigest, outAuthSetupParams)<br>Compare HM to resAuth. This verifies returnCode and output parameters. | tag<br>paramSize<br>returnCode<br>outArgOne<br>nonceEven<br>continueAuthSession<br>resAuth | ← | Return output parameters<br>Destroy auth session associated with authHandle |

The TPM MUST enable ADIP by using the OS-AP. The TPM MUST encrypt the authorization data for the new entity by performing an XOR using the shared secret created by the OS-AP.

The TPM MUST destroy the OS-AP session whenever a new entity is created.

**ADCP - Changing Authorization Data**

All entities from the Owner to the SRK to individual keys and data blobs have authorization data. This data may need to change at some point in time after the entity creation. The ADCP allows the entity owner to change the authorization data. The entity owner of a wrapped key is the owner of the parent key.

A requirement is that the owner must remember the old authorization data. The only mechanism to change the authorization data when the entity owner forgets the current value is to delete the entity and then recreate it.

To protect the data from exposure to eavesdroppers or other attackers, the authorization data uses the same encryption mechanism in use during the ADIP.

Changing authorization data requires opening two authentication handles. The first handle authenticates the entity owner (or parent) and the right to load the entity. This first handle is an OS-AP and supplies the data to encrypt the new authorization data according to the ADIP protocol. The second handle can be either an OI-AP or an OS-AP, it authorizes access to the entity for which the authorization data is to be changed.

The authorization data in use to generate the OS-AP shared secret must be the authorization data of the parent of the entity to which the change will be made.

When changing the authorization data for the SRK, the first handle OS-AP must be setup using the TPM Owner authorization data. This is because the SRK does not have a parent, per se.

If the SRKAuth data is known to userA and userB, userA can snoop on userB while userB is changing the authorization for a child of the SRK, and deduce the child's newAuth. Therefore, if SRKAuth is a well-known value, TPM_ChangeAuthAsymStart and TPM_ChangeAuthAsymFinish are preferred over TPM_ChangeAuth when changing authorization for children of the SRK.

This applies to all children of the SRK, including TPM identities.

The TSF permits the TSF, local or remote users to initiate communication via the trusted path. The TSF initiates communication only as a response to a command received via the trusted path. Local or remote users may initiate communication on the trusted LPC communication path by transmitting an approved TCG command ordinal inside the specified TCG command protocol, which is wrapped inside the standard LPC communications.

The TSF requires the user of the trusted path for initial user authentication, for all TPM commands, all user commands, and TSF responses. Communication by any other method, path or protocol is not recognized by the TSF and will initiate no response by the TSF.

### 6.1.8 Strength of Function Requirement

The strength of function specified for the TOE authentication function is SOF-Medium. The strength of cryptographic algorithms is outside the scope of the CC. Strength of function applies only to non-cryptographic, probabilistic or permutational mechanisms. The SOF requirement applies to the identification and authentication functionality within the TOE.

A SOF rating reflects the attacker, described in terms of attack potential, against which the probabilistic or permutational security function is designed to protect. To determine a SOF rating for the I&A functionality, the attack potential was calculated by the following method: Using Table B.3 from the CEM Annex B, a numerical score for attack potential was calculated and then Table B.4 from the CEM Annex B was used to translate the number into a qualitative attack potential and an SOF rating.

Table B.4 from CEM Annex B

| Range of Values | Resistant to attack with attack potential of: | SOF rating |
|---|---|---|
| <10 | No rating | No rating |
| 10 – 17 | Low | Basic |
| 18 – 24 | Moderate | Medium |
| >25 | High | High |

In the TOE, I&A data is known as “authorization data,” which is associated with each entity. The authorization data is a 160-bit field that is a SHA-1 hash of an eight-byte password. The TOE stores the authorization data in a “shielded location,” which is an area where data is protected against interference and prying, independent of its form. The entity owner has a copy of the data and protects the data according to procedures defined in the administrator and user guides. The authorization data is a shared secret between the TOE and the entity owner and is stored in the form of a data blob. Although there is a separate piece of authorization data for each entity, there is no requirement that each authorization data blob must be unique.

The TOE treats the authorization data as shielded data, an approach that requires that only protected capabilities access the authorization data.

In compliance with the Administrator and User Guide, the passwords:

- Must be exactly 8 characters long and must not be zeros.
- Must contain alphanumeric characters only.
- Must not be a common word, a word in any existing password dictionaries, or a word easily guessed (such as “password”).

Entity owner passwords are discretionary, however, in compliance with the Administrator and User Guide, users must use passwords and follow the same guidelines for selecting passwords as those of the administrator, listed above. However, since the use of the guidance does not represent the “worst case” scenario, it is assumed that entity owners will not follow password guidelines.

Analysis was performed using the following assumptions:

- It is assumed that attackers would have access to commonly available password crackers, particularly those that use dictionary and exhaustive search attacks.
- It is assumed that the environment provides protections such that passwords could not be captured en route to the chip, therefore the analysis covers only those attacks that guess

passwords or retrieve them from the TOE through some vulnerability; the scope of the SOF analysis is the TOE.

- It is assumed that the entity owners will not follow password guidelines in the Administrator and User Guide. Although words easily guessed or those in a known password dictionary are not supposed to be used, in order to present the worst-case scenario, it is assumed that entity owners will select natural language passwords, thereby greatly reducing the possible passwords that could be used.
- It is assumed that natural language passwords number no greater than 100,000.
- It is assumed that the attacker would first use a dictionary attack that would include common strategies for guessing passwords such as selecting a user login name, pAsSwOrD, simple transformations for common words, etc.
- Motivation of the attacker is not considered as part of this analysis because the system is multi-purpose and there is no way of knowing the value of the assets protected by the TOE. It is assumed that the value of the assets is low and therefore motivation on the part of the attacker is moderate to low.
- It is assumed that there are no time limitations on the attacker.

The chip does not allow reads to protected locations. Based on the vulnerability analysis, there is no obvious vulnerability such as buffer overflow, etc, that would allow an attacker access to these registers. Therefore, the SOF analysis focused on password “cracker” attacks. With 100,000 password possibilities, on average, the cracker would guess the password in 50,000 tries.

If there were no other protections, it would be relatively simple to break the password mechanism in a short time with 50,000 tries. However, the TOE has authentication failure protection, as described in Section 6.1.4, which locks the attacker out of the system after a specified number of failed attempts. Thereafter, every single subsequent failed attempt causes the lockout period to double. Given the authentication failure protection mechanism, it would take months to crack the password, given that the attacker on average must try 50,000 passwords and the lockout periods double with each lockout.

The attack potential for the TOE authentication mechanism was scored using Table B.3 in Annex B.8 of the CEM. The attack potential was scored as 18, i.e., a layman with standard equipment (score of 2) and Public knowledge of the TOE (score of 2 for both Identifying and Exploiting values) would take more than a month to guess the password, with the score of 8 for elapsed time and 6 for Access to the TOE (total of 18). The calculated attack potential is therefore 18, which exceeds requirements for SOF-Medium.

## 6.2 Assurance Measures

The assurance level selected for the TOE was EAL3 because EAL3 is applicable in those circumstances where users require a moderate level of independently assured security and require a thorough investigation of the TOE and its development without substantial re-engineering. EAL3 provides assurance by an analysis of the security functions, using a functional and interface specification, guidance documentation, and the high-level design of the TOE, to understand the security behavior.

EAL3 analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, strength of function analysis, and evidence of a developer search for obvious vulnerabilities.

Appropriate assurance measures are employed to satisfy the security assurance requirements. The TOE evaluation confirmed that the assurance measures are sufficient to satisfy the assurance requirements. The assurance measures are described in the set of evaluation evidence listed in Table 8, below. The documents listed in the table were used to satisfy assurance evaluation requirements.

*Table 8. Evaluation Evidence for Assurance Requirements*

| Assurance Requirement | Assurance Requirement Name | Evaluation Evidence | Rationale |
|---|---|---|---|
| ACM_CAP.3 | Authorization controls | Procedure Number 800-0012, Tracking Configuration Management Requirements document number 800-0012<br><br>Procedure Number 100-055, Bills of Material and Product Flow Requirements<br><br>Procedure Number 100-056, Part Number Generation<br><br>Procedure Number 800-008, Atmel Trusted Platform Module Data Sheet Distribution Procedure<br><br>Policy Number 7022, General Policy Specification for Process Control<br><br>Policy Number 7040, General Policy Specification for Document and Data Control<br><br>AT97SC3201 Pattern and Mask List, Specification Number AT56813Procedure Number 100-022, Wafer Fabrication Process Routes, Flows, And Pattern Mask Lists | This evidence was written to address the configuration management documentation for EAL3. This includes identifying the evaluated TOE and providing a configuration list with configuration items that have been uniquely identified and the method used to identify them. |
| ACM_SCP.1 | TOE CM coverage | Procedure Number 800-0012, Tracking Configuration Management Requirements document number 800-0012<br><br>AT97SC3201 Pattern and Mask List, Specification Number AT56813Procedure Number 100-022, Wafer Fabrication Process Routes, Flows, And Pattern Mask Lists | This evidence addresses CM coverage of the TOE. |
| ADO_DEL.1 | Delivery procedures | Procedure 793-300, Standard Operating Procedure for Shipping | This evidence addresses delivery procedures for the TOE and documents how the TOE is securely provided to the customer. |
| ADO_IGS.1 | Installation, generation, and start-up procedures | ATMEL Trusted Platform Module (AT97SC3201) Administrator and User Guide, Version 1.0, March 5, 2004 | This evidence addresses Installation, Generation, and Startup procedures for the evaluated TOE. This includes |

| | | | |
|---|---|---|---|
| | | | that the TOE is installed, generated, and started as the developers intended with the assurance that each time it is done the securely and the same way. |
| ADV_FSP.1 | Informal functional specification | Trusted Computing Group (TCG) Main Specification, version 1.1b<br><br>AT97SC3201 Datasheet, Atmel Trusted Platform Module, AT97SC3201 Advance Specification Summary, Rev.2015CX-07/17/01<br><br>ATMEL AT97SC3201 Trusted Platform Module Atmel-Specific Commands, Version 0.17, 4/12/02 | This evidence addresses the security functions of the TOE. This includes identifying and describing the external TOE security function interfaces. |
| ADV_HLD.2 | Security enforcing high-level design | Trusted Computing Group (TCG) Main Specification, version 1.1b<br><br>AT97SC3201 Datasheet, Atmel Trusted Platform Module, AT97SC3201 Advance Specification Summary, Rev.2015CX-07/17/01<br><br>ATMEL AT97SC3201 Trusted Platform Module Atmel-Specific Commands, Version 0.17, 4/12/02 | This evidence describes the security functionality of the TOE and supporting protection mechanisms implemented. |
| ADV_RCR.1 | Informal correspondence demonstration | Atmel AT97SC3201 Informal Correspondence Demonstration, Version 1.0 | This evidence was written specifically to show a correspondence analysis between the ST and the functional specification; between the functional specification and the high level design; and between the functional specification and the security policy model. |
| ADV_SPM.1 | Informal TOE security policy model | Atmel AT97SC3201 Security Policy Model, Version 1.0 | This evidence provides a security policy model for secure implementation and operation of the TOE. |
| AGD_ADM.1 | Administrator guidance | ATMEL Trusted Platform Module (AT97SC3201) Administrator and User Guide, Version 1.0, March 5, 2004 | This evidence addresses administrator guidance. It describes how to securely administer the TOE. |
| AGD_USR.1 | User guidance | ATMEL Trusted Platform Module (AT97SC3201) Administrator and User Guide, Version 1.0, March 5, 2004 | This evidence addresses user guidance. It describes the instructions and guidelines for secure use of the TOE. |
| ALC_DVS.1 | Identification of security measures | Procedure Number 800-011, Atmel Trusted Platform Module Lifecycle Management | This evidence provides the development security procedures in place. |
| ALC_FLR.1 | Basic flaw remediation | Policy 7026, General Quality Specification For Major Change Determination, Notification, And Response<br><br>ATMEL Trusted Platform Module (AT97SC3201) Administrator and User Guide, Version 1.0, March 5, 2004 | This evidence specifies procedures for basic flaw remediation. |
| ATE_COV.2 | Analysis of coverage | Master Verification Coverage Document, Version 1.9 for Atmel AT97SC3201 Trusted Platform Module | This evidence addresses the requirements for test coverage analysis evidence. This |

| | | | |
|---|---|---|---|
| | | | includes showing which security functions were tested. |
| ATE_DPT.1 | Testing: high-level design | Master Verification Coverage Document, Version 1.9 for Atmel AT97SC3201 Trusted Platform Module | This evidence shows the depth of testing for each of the functions. |
| ATE_FUN.1 | Functional testing | Master Verification Coverage Document, Version 1.9 for Atmel AT97SC3201 Trusted Platform Module | This evidence documents all functional tests. |
| ATE_IND.2 | Independent testing - sample | Not applicable | Not applicable, this function is performed by the evaluator |
| AVA_MSU.1 | Examination of guidance | ATMEL Trusted Platform Module (AT97SC3201) Administrator and User Guide, Version 1.0, March 5, 2004 | This evidence provides user and administrator guidance so that the evaluator can determine if misuse is possible based on the guidance. |
| AVA_SOF.1 | Strength of TOE security function evaluation | This Security Target | Included in the Security Target – specifies the SOF for I&A functionality. |
| AVA_VLA.1 | Developer vulnerability analysis | ATMEL AT97SC3201, Vulnerability Analysis, Version 1.0 | This evidence addresses the intended environment for the TOE and shows that there are no exploitable obvious vulnerabilities. |

# 7 PP Claims

This Security Target does not claim conformance with a PP, however, the ST is modeled on the Trusted Computing Platform Alliance (TCPA) Trusted Platform Module Protection Profile.

# 8 Rationale

This section provides further evidence and explanation to support the certification of this ST.

## 8.1 Security Objectives Rationale

Table 9 maps assumptions and threats to objectives, demonstrating that all assumptions and threats are mapped to at least one objective. Table 10 maps objectives to threats and assumptions, demonstrating that all objectives are mapped to at least one threat or assumption. A discussion of the rationale for threat mappings is provided below.

*Table 9. Mapping the TOE Security Environment to Objectives*

| # | Assumption/Threat | Objectives |
|---|---|---|
| 1E | A.Configuration | OE.Configuration |
| | | |
| 1 | T.Attack | O.DAC, O.I&A, O.Security_Roles, O.Self_Protect |
| 2 | T.Bypass | O.HMAC, O.Security_Attr_Mgt, O.Invoke |
| 3 | T.Export | O.Export |
| 4 | T.Hack_Crypto | O.Crypto_Op |
| 5 | T.I&A | O.I&A, O.Security_Roles, O.Export |
| 6 | T.Import | O.Import |
| 7 | T.Key_Gen_Destroy | O.Crypto_Key_Man |
| 8 | T.Malfunction | O.Fail_Secure |
| 9 | T.Modify | O.Limit_Actions_Auth, O.Security_Attr_Mgt, O.Security_Roles, O.DAC |
| 10 | T.Object_Attr_Default | O.Object_Attr_Default |
| 11 | T.Object_Attr_Change | O.Object_Attr_DefaultOver |
| 12 | T.Object_SecureValues | O.Obj_Attr_SecureValues |
| 13 | T.Residual_Info | O.No_Residual_Info, O.Crypto_Key_Man |
| 14 | T.Replay | O.Single_Auth |
| 15 | T.Repudiate_Transact | O.MessageNR |
| 16 | T.Test | O.Self_Test, O.General_Integ_Checks |

*Table 10. Tracing Security Objectives to Assumptions and Threats*

| # | Objectives | Assumptions/Threats |
|---|---|---|
| 1E | OE.Configuration | A.Configuration |
| | | |
| 1 | O.Crypto_Key_Man | T.Residual_Info, T.Key_Gen_Destroy |
| 2 | O.Crypto_Op | T.Hack_Crypto |
| 3 | O.Self_Test | T.Test |
| 4 | O.DAC | T.Attack, T.Modify |
| 5 | O.Export | T.Export, T.I&A |
| 6 | O.Fail_Secure | T.Malfunction |
| 7 | O.General_Integ_Checks | T.Test |
| 8 | O.HMAC | T.Bypass |
| 9 | O.I&A | T.Attack, T.I&A |
| 10 | O.Import | T.Import |
| 11 | O.Invoke | T.Bypass |
| 12 | O.Limit_Actions_Auth | T.Modify |
| 13 | O.MessageNR | T.Repudiate_Transact |
| 14 | O.No_Residual_Info | T.Residual_Info |
| 15 | O.Object_Attr_Default | T.Object_Attr_Default |
| 16 | O.Object_Attr_DefaultOver | T.Object_Attr_Change |
| 17 | O.Obj_Attr_SecureValues | T.Object_SecureValues |
| 18 | O.Security_Attr_Mgt | T.Modify, T.Bypass |
| 19 | O.Security_Roles | T.Attack, T.Modify, T.I&A |
| 20 | O.Self_Protect | T.Attack |
| 21 | O.Single_Auth | T.Replay |

### 8.1.1 Threats

This section describes each threat and enumerates and discusses the security objectives that counter the threat.

**T.Attack**: An undetected compromise of the cryptography-related IT assets may occur as a result of an attacker (whether an insider or outsider) attempting to perform actions that the individual is not authorized to perform.

T.Attack is countered by O.DAC, O.I&A, O.Security_Roles, and O.Self_Protect. These objectives limit the ability of a user to the performance of only those actions that the user is authorized to perform:

- O.DAC: The TOE shall provide its users with the means of controlling and limiting access to the TOE assets in accordance with a specified access control policy. This objective limits an attacker from performing unauthorized actions through a defined access control policy.
- O.I&A: The TOE shall uniquely identify all users, and shall authenticate the claimed identify before granting a user access to the TOE facilities. This objective supports the access control policy by uniquely identifying users (key pairs within the TOE) so that specific access control rules can be applied for each user role.
- O.Security_Roles: The TOE shall maintain security-relevant roles and association of users with those roles. This objective further supports the access control policy by associating each user with a role, which then can be assigned a specific access control policy.
- O.Self_Protect: The TSF will maintain a domain for its own execution that protects it and its resources from external interference, tampering, or unauthorized disclosure.

**T.Bypass**: An unauthorized individual or user may tamper with security attributes or other data in order to bypass TOE security functions and gain unauthorized access to TOE assets.

T.Bypass is countered by O.HMAC, O.Security_Attr_Mgt, and O.Invoke. These three objectives allow the TOE to detect tampering with data and to counter the ability of unauthorized users from tampering with security attributes or other data:

- O.HMAC: The TOE shall provide the ability to detect the modification of security attributes and other data. This objective provides the capability for the system to detect tampering with data.
- O.Security_Attr_Mgt: The TOE shall allow only authorized users to initialize and change object security attributes. This objective requires that only authorized users be allowed to initialize and change security attributes, which counters the threat of an unauthorized user making such changes.
- O.Invoke: The TSF shall be invoked for all actions. This objective assists in the protection of the system from tampering by unauthorized users, since it requires the TSF to be invoked for all actions and does not allow it to be bypassed by any user.

**T.Export**: A user or an attacker may export data without security attributes or with insecure security attributes, causing the data exported to be erroneous and unusable, to allow erroneous data to be added or substituted for the original data, and/or to reveal secrets.

T.Export is countered by O.Export. O.Export states: When data are exported outside the TPM, the TOE shall ensure that the data security attributes being exported are unambiguously associated with the data.

**T.Hack_Crypto**: Cryptographic algorithms may by incorrectly implemented, allowing an unauthorized individual or user to decipher keys generated within the TPM and thereby gain unauthorized access to encrypted data.

T.Hack_Crypto is countered by O.Crypto_Op, which states: The TOE shall perform cryptographic operations, including secure hash, random number generation, HMAC, RSA digital signature and signature verification, RSA encryption and decryption, and RSA key generation in accordance with specified algorithms and key size; key size must be sufficient size to minimize the risk of deciphering private/public key pairs.

**T.I&A**: An authorized or unauthorized TOE user may gain access to TOE data, keys, and operations to which they are not authorized access.

T.I&A is countered by O.I&A, O.Security_Roles, and O.Import. These objectives require a user to be identified and authenticated and to function under a predefined role with specified access control policy:

- O.I&A: The TOE shall uniquely identify all users, and shall authenticate the claimed identify before granting a user access to the TOE facilities that require authorization. This objective requires identification and authentication of users (key pairs within the TOE) so that specific access control rules can be applied for each user role.
- O.Security_Roles: The TOE shall maintain security-relevant roles and association of users with those roles. This objective further requires the association of each user with a role, which then can be assigned a specific access control policy.
- O.Export: When data are exported outside the TPM, the TOE shall ensure that the data security attributes being exported are unambiguously associated with the data.

**T.Import**: An authorized or unauthorized user may import data or keys without security attributes or with erroneous security attributes, causing key ownership and authorization to be uncertain or erroneous and the system to malfunction or operate in an insecure manner.

T.Import is countered by O.Import, which states: When data are being imported into the TOE, the TOE shall ensure that the data security attributes are being imported with the data and the data is from authorized source. In addition, the TOE shall verify those security attributes according to the TSF access control rules.

**T.Key_Gen_Destroy**: Cryptographic keys may be generated or destroyed in an insecure manner, causing key compromise.

T.Key_Gen_Destroy is countered by O.Crypto_Key_Man, which states: The TOE shall generate and destroy cryptographic keys in a secure manner.

**T.Malfunction**: TOE assets may be modified or disclosed to an unauthorized individual or user of the TOE, through malfunction of the TOE.

T.Malfunction is countered by O.Fail_Secure, which states: The TOE shall preserve the secure state of the system in the event of a cryptographic or other failure.

**T.Modify**: An attacker may modify data, e.g., stored security attributes or keys, in order to impersonate an authorized user or to gain access to the TOE and its assets. The integrity of the information may be compromised due to the unauthorized modification or destruction of the information by an attacker.

T.Modify is countered by O.Limit_Actions_Auth, O.Security_Attr_Mgt, O.Security_Roles, and O.DAC. These objectives support the ability of the TOE to limit unauthorized user access and to maintain data and system integrity through appropriate management of cryptographic data in particular:

- O.Limit_Actions_Auth: The TOE shall restrict the actions a user may perform before the TOE verifies the identity of the user.
- O.Security_Attr_Mgt: The TOE shall allow only authorized users to initialize and change object security attributes.
- O.Security_Roles: The TOE shall maintain security-relevant roles and association of users with those roles.
- O.DAC: The TOE shall control and restrict user access to the TOE assets in accordance with a specified access control policy.

**T. Object_Attr_Default**: An attacker may create an object with no security attribute values.

T.Object_Attr_Default is countered by O.Object_Attr_Default, which states: The TOE shall require default security attributes for an object when an object is created.

**T.Object_Attr_Change**: A user or attacker may make unauthorized changes to security attribute values for an object.

T.Object_Attr_Change is countered by O.Object_Attr_DefaultOver, which states: The TOE shall permit authorized users to override defaulted values for security attributes for an object.

**T.Object_SecureValues**: An attacker or user may set unsecured values for object security attributes.

T.Object_SecureValues is countered by O.Obj_Attr_SecureValues, which states: The TOE shall maintain object security attributes by permitting only secure values; secure values are security parameters associated with a key that require owner authorization.

**T.Residual_Info**: A user may obtain information that the user is not authorized to have when the data is no longer actively managed by the TOE ("data scavenging").

T.Residual_Info is countered by O.No_Residual_Info and O.Crypto_Key_Man. O.No_Residual_Info ensure that no residual data is left in buffers or system locations. O.Crypto_Key_Man specifies that cryptographic key destruction must be performed:

- O.No_Residual_Info: The TOE shall ensure there is no "object reuse," i.e., ensure that there is no residual information in information containers or system resources upon their reallocation to different users.
- O.Crypto_Key_Man: The TOE shall generate and destroy cryptographic keys in a secure manner.

**T.Replay**: An unauthorized individual may gain access to the system and sensitive data through a "replay" attack that allows the individual to capture identification and authentication data.

T.Replay is countered by O.Single_Auth, which states: The TOE shall provide a single use authentication mechanism and require re-authentication to prevent "replay" attacks.

**T.Repudiate_Transact**: An originator of data may deny originating the data to avoid accountability.

T.Repudiate_Transact is countered by O.MessageNR, which states: The TOE shall provide user data integrity, source authentication, and the basis for source non-repudiation when exchanging data with a remote system.

**T.Test**: The TOE may start-up in an insecure state or enter an insecure state, allowing an attacker to obtain sensitive data or compromise the system.

T.Test is countered by O.Self_Test and O.General_Integ_Checks. These objectives require the TOE to provide self-test and integrity checking functionality in order to detect unsecured states either at startup or during normal operation:

- O.Self_Test: The TOE shall provide the ability to verify that the TSF functions operate as designed.
- O.General_Integ_Checks: The TOE shall provide periodic integrity checks on both system and user data.

## 8.2 Security Requirements Rationale

In this section, the objectives are mapped to the functional requirements and rationale is provided for the selected EAL and its components and augmentation.

### 8.2.1 Functional Security Requirements Rationale

The mapping of security objectives to functional requirements (components) is provided in Table 11.

*Table 11. Mapping Security Objectives to Functional Components*

| | **Objectives** | **Functional Component** |
|---|---|---|
| 1 | O.Crypto_Key_Man | FCS_CKM.1, FCS_CKM.4 |
| 2 | O.Crypto_Op | FCS_CKM.1, FCS_COP.1 (all iterations) |
| 3 | O.Self_Test | FPT_AMT.1 |
| 4 | O.DAC | FDP_ACC.1, FDP_ACF.1, FMT_MOF.1, FMT_MTD.1 (all iterations) |
| 5 | O.Export | FDP_ETC.2 |
| 6 | O.Fail_Secure | FPT_FLS.1, FPT_RCV.4 |
| 7 | O.General_Integ_Checks | FPT_TPMTST.1, FPT_AMT.1 |
| 8 | O.HMAC | FCS_COP.1:4 |
| 9 | O.I&A | FIA_UAU.1, FIA_UID.1, FIA_ATD.1 |
| 10 | O.Import | FDP_ITC.2, FPT_TDC.1, FTP_TRP.1 |
| 11 | O.Invoke | FPT_RVM.1 |
| 12 | O.Limit_Actions_Auth | FIA_UAU.1, FIA_UID.1 |
| 13 | O.MessageNR | FCO_NRO.2, FDP_ETC.2 |
| 14 | O.No_Residual_Info | FDP_RIP.2 |
| 15 | O.Object_Attr_Default | FMT_MSA.3 |
| 16 | O.Object_Attr_DefaultOver | FMT_MSA.3 |
| 17 | O.Obj_Attr_SecureValues | FMT_MSA.2, FPT_TDC.1 |
| 18 | O.Security_Attr_Mgt | FMT_MSA.3, FMT_MSA.1 |
| 19 | O.Security_Roles | FMT_SMR.2, FIA_ATD.1 |
| 20 | O.Self_Protect | FPT_SEP.1 |
| 21 | O.Single_Auth | FIA_UAU.4, FIA_UAU.6, FPT_RPL.1 |

A discussion of the rationale for the mapping is provided for each objective below.

**O.Crypto_Key_Man**: The TOE shall generate and destroy cryptographic keys in a secure manner.

O.Crypto_Key_Man is mapped to:

- FCS_CKM.1, Cryptographic key generation, which requires that cryptographic keys be generated in accordance with the RSA algorithm with specified cryptographic sizes that meet PKCS #1 V.2 standard.
- FCS_CKM.4, Cryptographic key destruction, which requires that cryptographic keys be destroyed in accordance with a specified secure key destruction method.

**O.Crypto_Op**: The TOE shall perform cryptographic operations, including secure hash, random number generation, HMAC, RSA digital signature and signature verification, RSA encryption and decryption, and RSA key generation in accordance with specified algorithms and key size; key size must be sufficient size to minimize the risk of deciphering private/public key pairs.

O.Crypto_Op is mapped to:

- FCS_CKM.1, Cryptographic key generation, which requires that cryptographic keys be generated in accordance with the RSA algorithm with specified cryptographic sizes that meet PKCS #1 V.2 standard.
- FCS_COP.1, Cryptographic operations. There are four iterations of this component, including RSA encrypt and decrypt, RSA signature and signature verification, random number generation, SHA, and Keyed-Hashing for Message Authentication. The iterations cover all cryptographic operations and specify key sizes and standards that must be met.

**O.Self_Test**: The TOE shall provide the ability to verify that the TSF functions operate as designed.

O.Self_Test is mapped to:

- FPT_AMT.1, Abstract machine testing. This component tests the cryptographic portion of the underlying abstract state machine.

**O.DAC**: The TOE shall provide its users with the means of controlling and limiting access to the TOE assets in accordance with a specified access control policy.

O.DAC is mapped to:

- FDP_ACC.1, Subset access control, which requires that Protected Operations Access Controls by enforced on subjects, objects and operations.
- FDP_ACF.1, Security attribute based access control, which defines access controls based on TCPA_AUTH_DATA_USAGE and TCPA_KEY_USAGE values.
- FMT_MOF.1, Management of security functions behavior, allows the ST author to specify the list of functions that are restricted to the TPM owner.
- FMT_MTD.1, Management of TSF Data, ensures that the TSF data is accessible to authorized users.

**O.Export**: When data are exported outside the TPM, the TOE shall ensure that the data security attributes being exported are unambiguously associated with the data.

O.Export is mapped to:

- FDP_ETC.2, Export of user data with security attributes, which requires that data exported outside the TSF have security attributes that are unambiguously associated with the data exported.

**O.Fail_Secure**: The TOE shall preserve the secure state of the system in the event of a cryptographic or other failure.

O.Fail_Secure is mapped to:

- FPT_FLS.1, Failure with preservation of secure state, which requires that the TSF preserve a secure state in the event of a failure.
- FPT_RCV.4, Function recovery, which requires that all TPM Commands either complete successfully or fail and recover to a secure state.

**O.General_Integ_Checks**: The TOE shall provide periodic integrity checks on both system and user data.

O.General_Integ_Checks is mapped to:

- FPT_AMT.1: Abstract machine testing. This component tests the cryptographic portion of the underlying abstract state machine.
- FPT_TPMTST.1, TPM integrity test. This component requires the TSF to run a suite of self-tests during initial start-up and at the request of an authorized user to demonstrate the correct operation of the TSF.

**O.HMAC**: The TOE shall provide the ability to detect the modification of security attributes and other data.

O.HMAC is mapped to:

- FCS_COP.1.1;4, which requires that the TOE provide HMAC capability in conformance with the referenced standard to provide the ability to detect the modification of security attributes and other data.

**O.I&A**: The TOE shall uniquely identify all users, and shall authenticate the claimed identify before granting a user access to the TOE facilities that require authorization. The TPM identification and authentication capability is used to authenticate an entity owner and to authorize use of an entity. The basic premise is to prove knowledge of a shared secret. This shared secret is the identification and authentication data. Note that the TCG Main Specification document refers to the identification and authentication process and this data as *authorization*.

O.I&A is mapped to:

- FIA_UAU.1, Timing of authentication, which states that a user shall be successfully authenticated before performing all actions except those explicitly defined.
- FIA_UID.1, Timing of identification, which states that a user shall be successfully identified before performing all actions except those explicitly defined.
- FIA_ATD.1, User attribute definition, which supports FIA_UAU.1 and FIA_UID.1 by providing a requirement for user attributes. Authentication data is defined as a user attribute. Authentication data in this case is associated with a specific key, which is analogous to a user.

**O.Import**: When data are being imported into the TOE, the TOE shall ensure that the data security attributes are being imported with the data and the data is from authorized source. In addition, the TOE shall verify those security attributes according to the TSF access control rules.

O.Import is mapped to:

- FDP_ITC.2, Import of user data with security attributes, which states that data imported into the TOE must have security attributes. These include authentication data on user keys.
- FPT_TDC.1, Inter-TSF basic TSF data consistency, defines security attributes and requires that they be consistently interpreted when importing data.
- FTP_TRP.1, Trusted path ensures that the data is being received from an authorized source. Trusted path is also a dependency of FDP_ITC.2, requiring a trusted path for data import.

**O.Invoke**: The TSF shall be invoked for all actions.

O.Invoke is mapped to:

- FPT_RVM.1, Non-bypassability of the TSP, which ensures that TSP functions are invoked and succeed before each function within the TSC is allowed to proceed.

**O.Limit_Actions_Auth**: The TOE shall restrict the actions a user may perform before the TOE verifies the identity of the user.

O.Limit_Actions_Auth is mapped to:

- FIA_UAU.1, Timing of authentication, which states that a user shall be successfully authenticated before performing all actions except those explicitly defined.
- FIA_UID.1, Timing of identification, which states that a user shall be successfully identified before performing all actions except those explicitly defined.

**O.MessageNR**: The TOE shall provide user data integrity, source authentication, and the basis for source non-repudiation when exchanging data with a remote system.

O.MessageNR is mapped to:

- FCP_NRO.2, Enforced proof of origin, which requires that the TSF enforce generation of data that provides evidence of origin for data transmitted.
- FDP_ETC.2, Export of user data with security attributes, ensures that access control SFPs and security attributes are associated with exported data, thereby providing user data integrity.

**O.No_Residual_Info**: The TOE shall ensure there is no “object reuse,” i.e., ensure that there is no residual information in information containers or system resources upon their reallocation to different users.

O.No_Residual_Info is mapped to:

- FDP_RIP.2, Full residual information protection, which requires that any previous information content of a resource be made unavailable.

**O.Object_Attr_Default**: The TOE shall require default security attributes for an object when an object is created.

O.Object_Attr_Default is mapped to:

- FMT_MSA.3, Static attribute initialization, which requires that security attributes be specified and that certain defaults be in place.

**O.Object_Attr_DefaultOver**: The TOE shall permit authorized users to override defaulted values for security attributes for an object.

O.Object_Attr_DefaultOver is mapped to:

- FMT_MSA.3, Static attribute initialization, which requires that security attributes be specified, that certain defaults be defined, and that authorized users have the capability to override the defaults.

**O.Obj_Attr_SecureValues**: The TOE shall maintain object security attributes by permitting only secure values; secure values are security parameters associated with a key that require owner authorization.

O.Obj_Attr_SecureValues is mapped to:

- FMT_MSA.2, Secure security attributes, which requires that only secure values be accepted for security attributes.
- FPT_TDC.1, Inter-TSF basic TSF data consistency, defines security attributes.

**O.Security_Attr_Mgt**: The TOE shall allow only authorized users to initialize and change object security attributes.

O.Security_Attr_Mgt is mapped to:

- FMT_MSA.3, Static attribute initialization, which requires that security attributes be specified.
- FMT_MSA.1, Management of security attributes, which specifies that access controls requiring security attributes for objects be enforced.

**O.Security_Roles**: The TOE shall maintain security-relevant roles and association of users with those roles.

O.Security_Roles is mapped to:

- FMT_SMR.2, Restrictions on security roles, which requires that the TSF maintain roles and that the roles be associated with users.
- FIA_ATD.1, User attribute definition, which provides a requirement for user attributes. Authentication data is defined as a user attribute. Authentication data in this case is associated with a specific key, which is analogous to a user. Note that the TPM identification and authentication capability is used to authenticate an entity owner and to authorize use of an entity. The basic premise is to prove knowledge of a shared secret. This shared secret is the identification and authentication data. Note that the TCG Main Specification document refers to the identification and authentication process and this data as *authorization*.

**O.Self_Protect**: The TSF will maintain a domain for its own execution that protects it and its resources from external interference, tampering, or unauthorized disclosure.

O.Self_Protect is mapped to FPT_SEP.1, TSP domain separation which requires the TSF to protect itself.

**O.Single_Auth**: The TOE shall provide a single use authentication mechanism and require re-authentication to prevent “replay” attacks.

O.Single_Auth is mapped to:

- FIA_UAU.4, Single-use authentication mechanisms, which prevents the reuse of authentication data.
- FIA_UAU.6, Re authenticating, which requires that a user be re authenticated for every command that requires user authentication.
- FPT_RPL.1, Replay detection, prevents replay attacks.

### 8.2.2 Assurance Requirement Rationale

EAL3 was selected because the TOE requires a moderate level of independently assured security and requires a thorough investigation of the TOE and its development without substantial re-engineering. EAL3 provides assurance by an analysis of the security functions, using a functional and interface specification, guidance documentation, and the high-level design of the TOE to understand the security behaviour. The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, strength of function analysis, and evidence of a developer search for obvious vulnerabilities.

EAL3 is augmented with ADV_SPM.1 because ADV_SPM.1 is a dependency of functional security requirements FMT_MSA.2, FPT_FLS.1, and FPT_RCV.4. EAL3 is also augmented with ALC_FLR.1 to track and correct the reported and found security flaws in the product.

The assurance requirements are met by documentation and procedures specified in Section 6.2, Table 8, and are not repeated here.

### 8.2.3 Strength of Function Rationale

The TOE is designed to protect against “moderate” attack potential. Thus, based on the CEM Annex B, Table B.2, the strength of function is SOF Medium. The strength of cryptographic algorithms is outside the scope of the CC. Strength of function only applies to non-cryptographic, probabilistic or permutational mechanisms. The SOF requirement for the TOE, therefore, applies to the identification and authentication functionality for the TPM. The SOF analysis is provided in Section 6.1.8.

### 8.2.4 Dependency Rationale

Table 12 lists the Functional Requirements and their dependencies

*Table 12. Functional Requirements Dependencies*

| # | Requirement | Dependencies |
|---|---|---|
| 1 | FCO_NRO.2 | FIA_UID.1 |
| 2 | FCS_CKM.1 | FCS_COP.1, FCS_CKM.4, FMT_MSA.2 |
| 3 | FCS_CKM.4 | FCS_CKM.1, FMT_MSA.2 |
| 4 | FCS_COP.1 | FCS_CKM.1, FCS_CKM.4, FMT_MSA.2 |
| 5 | FDP_ACC.1 | FDP_ACF.1 |
| 6 | FDP_ACF.1 | FDP_ACC.1, FMT_MSA.3 |
| 7 | FDP_ETC.2 | FDP_ACC.1 |
| 8 | FDP_ITC.2 | FDP_ACC.1, FTP_TRP.1, FPT_TDC.1 |
| 9 | FDP_RIP.2 | None |
| 10 | FIA_ATD.1 | None |
| 11 | FIA_UAU.1 | FIA_UID.1 |
| 12 | FIA_UAU.4 | None |
| 13 | FIA_UAU.6 | None |
| 14 | FIA_UID.1 | None |
| 15 | FMT_MOF.1 | FMT_SMR.2 |

| # | Requirement | Dependencies |
|---|---|---|
| 16 | FMT_MSA.1 | FDP_ACC.1, FMT_SMR.2 |
| 17 | FMT_MSA.2 | ADV_SPM.1, FDP_ACC.1, FMT_MSA.1, FMT_SMR.2 |
| 18 | FMT_MSA.3 | FMT_MSA.1, FMT_SMR.2 |
| 19 | FMT_MTD.1 | FMT_SMR.2 |
| 20 | FMT_SMR.2 | FIA_UID.1 |
| 21 | FPT_AMT.1 | None |
| 22 | FPT_FLS.1 | ADV_SPM.1 |
| 23 | FPT_RCV.4 | ADV_SPM.1 |
| 24 | FPT_RPL.1 | None |
| 25 | FPT_RVM.1 | None |
| 26 | FPT_SEP.1 | None |
| 27 | FPT_TDC.1 | None |
| 28 | FPT_TMPTST.1 | None |
| 29 | FPT_TRP.1 | None |

### 8.2.5 Security Functional Requirements Grounding in Objectives

*Table 13. Requirements to Objectives Mapping*

| # | Requirements | Objectives |
|---|---|---|
| 1 | FCO_NRO.2 | O.MessageNR |
| 2 | FCS_CKM.1 | O.Crypto_Key_Man, O.Crypto_Op |
| 3 | FCS_CKM.4 | O.Crypto_Key_Man |
| 4-1 | FCS_COP.1; 1 | O.Crypto_Op |
| 4-2 | FCS_COP.1; 2 | O.Crypto_Op |
| 4-3 | FCS_COP.1; 3 | O.Crypto_Op |
| 4-4 | FCS_COP.1; 4 | O.Crypto_Op, O.HMAC |
| 4-5 | FCS_COP.1:5 | O.Crypto_Op |
| 5 | FDP_ACC.1 | O.DAC |
| 6 | FDP_ACF.1 | O.DAC |
| 7 | FDP_ETC.2 | O.Export, O.MessageNR |
| 8 | FDP_ITC.2 | O.Import |
| 9 | FDP_RIP.2 | O.No_Residual_Info |
| 10 | FIA_ATD.1 | O.I&A, O.Security_Roles |
| 11 | FIA_UAU.1 | O.I&A, O.Limit_Actions_Auth |
| 12 | FIA_UAU.4 | O.Single_Auth |
| 13 | FIA_UAU.6 | O.Single_Auth |
| 14 | FIA_UID.1 | O.I&A, O.Limit_Actions_Auth |
| 15 | FMT_MOF.1 | O.DAC |
| 16 | FMT_MSA.1 | O.Security_Attr_Mgt |
| 17 | FMT_MSA.2 | O.Obj_Attr_SecureValues |
| 18 | FMT_MSA.3 | O.Security_Attr_Mgt, O.Object_Attr_Default, O.Object_Attr_DefaultOver |
| 19 | FMT_MTD.1 | O.DAC (all iterations of FMT_MTD.1) |
| 20 | FMT_SMR.2 | O.Security_Roles |
| 21 | FPT_AMT.1 | O.Self_Test |
| 22 | FPT_FLS.1 | O.Fail_Secure |
| 23 | FPT_RCV.4 | O.Fail_Secure |
| 24 | FPT_RPL.1 | O.Single_Auth |
| 25 | FPT_RVM.1 | O.Invoke |
| 26 | FPT_SEP.1 | O.Self_Protect |
| 27 | FPT_TDC.1 | O.Obj_Attr_SecureValues, O.Import |
| 28 | FPT_TPMTST.1 | O.General_Integ_Checks |
| 29 | FTP_TRP.1 | O.Import |

### 8.2.6 Rationale for explicitly stated requirement

One explicitly stated requirement is included in this ST: FPT_TPMTST.1, TPM integrity test. This requirement was included to clearly define the requirement for the TSF to run a suite of self-tests during initial start-up of the TPM and at the request of a user to demonstrate the correct operation of the TSF. This functionality, particularly the requirement for a capability for self-tests to be run at the request of a user during operation, is required to meet the TCG Specification requirements. CC functional requirements were considered prior to defining an explicitly stated requirement. Although the Part 2 functional requirement FPT_TST.1, TSF testing, was considered, a portion of that requirement, FPT_TST.1.3, was not applicable to the TOE. No other requirements were found to be applicable; therefore, an explicitly stated requirement was created.

# Appendix – Acronyms and Glossary

## Acronyms

| | |
|---|---|
| **CC** | Common Criteria |
| **CRC** | Cyclic Redundancy Check |
| **CRT** | Chinese Remainder Theorem |
| **EAL** | Evaluation Assurance Level |
| **HMAC** | Hashing Message Authentication Code |
| **IT** | Information Technology |
| **LPC** | Low Pin Count |
| **PCR** | Platform Configuration Register |
| **PP** | Protection Profile |
| **SF** | Security Function |
| **SFP** | Security Function Policy |
| **SOF** | Strength of Function |
| **ST** | Security Target |
| **TOE** | Target of Evaluation |
| **TPM** | Trusted Platform Module |
| **TSC** | TSF Scope of Control |
| **TSF** | TOE Security Function |
| **TSFI** | TSF Interface |
| **TSP** | TOE Security Policy |

## Glossary

| | |
|---|---|
| **3DES** | DES using a key of a size that is 3X the size that of a DES key. See DES. |
| **Blob** | Opaque data of fixed or variable size. The meaning and interpretation of the data is outside the scope and context of the Subsystem. |
| **Challenger** | An entity that requests and has the ability to interpret integrity metrics from a Subsystem. |
| **Conformance Credential** | A credential that states the conformance to the TCG specification of: the TPM; the method of incorporation of the TPM into the platform; the RTM; and the method of incorporation of the RTM into the platform. |
| **Denial-of-service attack** | An attack on a system (or subsystem), which has no affect on information except to prevent its use. |
| **DES** | Symmetric key encryption using a key size of 56 bits defined by NIST as FIPS 46-3. |
| **Endorsement Credential** | A credential containing a public key (the endorsement public key) that was generated by a genuine TPM. |

| | |
|---|---|
| **Endorsement Key** | A term used ambiguously, depending on context, to mean a pair of keys, or the public key of that pair, or the private key of that pair; an asymmetric key pair generated by or inserted in a TPM that is used as proof that a TPM is a genuine TPM; the public endorsement key (PUBEK); the private endorsement key (PRIVEK). |
| **Identity Credential** | A credential issued by a Privacy CA that provides an identity for the TPM. |
| **Integrity metric(s)** | Values that are the results of measurements on the integrity of the platform. |
| **Man-in-the-middle attack** | An attack by an entity intercepting communications between two others without their knowledge and by intercepting that communication is able to obtain or modify the information between them. |
| **Migratable:** | A key that may be transported outside the specific TPM. |
| **Nonce** | A nonce is a random value that provides protection from replay and other attacks. Many of the commands and protocols in the specification require a nonce. |
| **Non-Migratable** | A key that cannot be transported outside a specific TPM; a key that is (statistically) unique to a particular TPM. |
| **Owner** | The entity that owns the platform in which a TPM is installed. Since there is, by definition, a one-to-one relationship between the TPM and the platform, the Owner is also the Owner of the TPM. The Owner of the platform is not necessarily the "user" of the platform (e.g., in a corporation, the Owner of the platform might be the IT department while the user is an employee.) The Owner has administration rights over the TPM. |
| **PKI Identity Protocol** | The protocol used to insert anonymous identities into the TPM. |
| **Platform Credential** | A credential that states that a specific platform contains a genuine TCG Subsystem. |
| **Privacy CA** | An entity that issues an Identity Credential for a TPM based on trust in the entities that vouch for the TPM via the Endorsement Credential, the Conformance Credential, and the Platform Credential. |
| **Private Endorsement Key (PRIVEK)** | The private key of the key pair that proves that a TPM is a genuine TPM. The PRIVEK is (statistically) unique to only one TPM. |
| **Public Endorsement Key (PUBEK)** | A public key that proves that a TPM is a genuine TPM. The PUBEK is (statistically) unique to only one TPM. |
| **Random number generator (RNG)** | A pseudo-random number generator that must be initialized with unpredictable data and provides "random" numbers on demand. |
| **Root of Trust for Measurement (RTM)** | The point from which all trust in the measurement process is predicated. |
| **Root of Trust for Reporting (RTR)** | The point from which all trust in reporting of measured information is predicated. |
| **Root of Trust for Storing (RTS)** | The point from which all trust in Protected Storage is predicated. |
| **RSA** | An (asymmetric) encryption method using two keys: a private key and a public key. Reference: http://www.rsa.com. |
| **SHA-1** | A NIST defined hashing algorithm producing a 160-bit result from an arbitrary sized source as specified in FIPS 180-1. |

| | |
|---|---|
| **Storage Root Key (SRK)** | The root key of a hierarchy of keys associated with a TPM; generated within a TPM; a non-migratable key. |
| **Subsystem** | The combination of the TSS and the TPM. |
| **Support Services (TSS)** | Services to support the TPM but which do not need the protection of the TPM. The same as Trusted Platform Support Services. |
| **TCG-protected capability** | A function that is protected within the TPM, and has access to TPM secrets. |
| **TPM Identity** | One of the anonymous PKI identities belonging to a TPM; a TPM may have multiple identities. |
| **Trusted Platform Agent (TPA)** | Trusted Platform Agent; the component within the platform that reports integrity metrics, logs, Validation Data, etc. to a Challenger; outside the scope of this specification. |
| **Trusted Platform Measurement Store (TPMS)** | Storage locations within the Subsystem, which contain unprotected logs of measurement process. |
| **Trusted Platform Module (TPM)** | The set of functions and data that are common to all types of platform, which must be trustworthy if the Subsystem is to be trustworthy; a logical definition in terms of protected capabilities and shielded locations. |
| **Trusted Platform Support Services (TSS)** | The set of functions and data that are common to all types of platform, which are not required to be trustworthy (and therefore do not need to be part of the TPM). |
| **User** | An entity that uses the platform in which a TPM is installed. The only rights that a User has over a TPM are the rights given to the User by the Owner. These rights are expressed in the form of authentication data, given by the Owner to the User that permits access to entities protected by the TPM. The User of the platform is not necessarily the "owner" of the platform (e.g., in a corporation, the owner of the platform might be the IT department while the User is an employee). There can be multiple Users. |
| **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. |
| **Validation Data** | Data inside a Validation Credential; the values that the integrity measurements should produce when the part of a platform described by the Validation Credential is working correctly. |
| **Validation Entity** | An entity that issues a Validation Certificate for a component; the manufacturer of that component; an agent of the manufacturer of that component. |