# TCG Specification Architecture Overview

**Specification**
**Revision 1.4**
**2nd August 2007**

**Contact:** admin@trustedcomputinggroup.org

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

Copyright © TCG 2003

TCG

Copyright © 2003 Trusted Computing Group, Incorporated.

**Disclaimer**

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

No license, express or implied, by estoppel or otherwise, to any TCG or TCG member intellectual property rights is granted herein.

**Except that a license is hereby granted by TCG to copy and reproduce this specification for internal use only.**

Contact the Trusted Computing Group at [website link] for information on specification licensing through membership agreements.

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



IPR2025-01572
Apple EX1048 Page 2

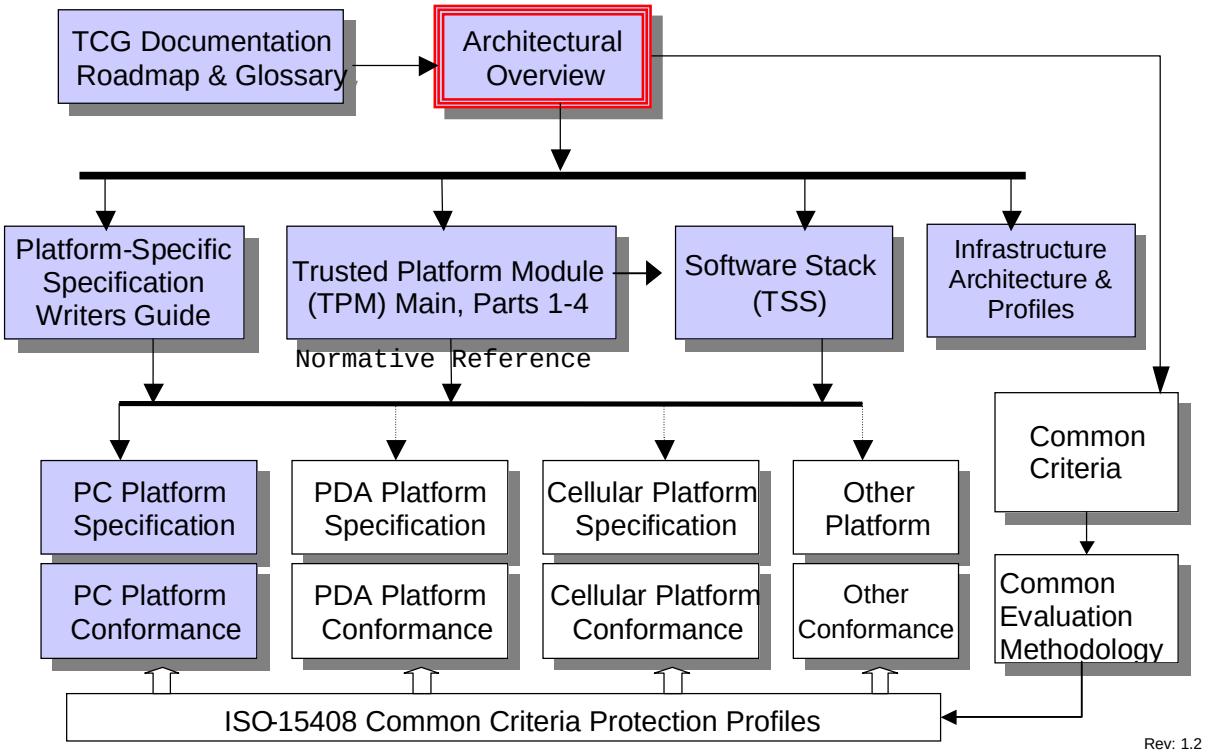
## Revision History

| | |
|---|---|
| 0.10 | Started: 1 April 2003. Last Updated: 04/30/01 by David Grawrock |
| 0.11 | Incorporated TC feedback. 11/19/03 – NMS |
| 0.13 | Merged changes from 0.12 and additional TC feedback |
| 0.14 | Spelling, grammar and other minor changes. |
| 0.15 | Applied TC feedback |
| 0.99 | Formatting and final review |
| 1.0 | Updated version, draft and TCG confidential markings. |
| 1.1 | Updated use case examples |
| 1.2 | Fixed stale terminology references in figures. |
| 1.3 | Corrected section 4.1.3 (remove “integrity storage”, add “integrity logging”) |
| 1.4 | Corrected section 5.3 (Common Criteria is ISO-15408, not 14508) |



IPR2025-01572
Apple EX1048 Page 3

# Document Roadmap




This document is identified by triple-line boarder in the document roadmap diagram.



IPR2025-01572
Apple EX1048 Page 4

# Table of Contents

1. Scope and Audience ... 1
2. What is TCG? ... 2
2.1 History ... 2
2.2 Mission ... 2
2.3 Goals ... 2
3. TCG Usage Scenarios ... 3
3.1 Risk Management ... 3
3.2 Asset Management ... 3
3.3 E-commerce ... 3
3.4 Security Monitoring and Emergency Response ... 3
4. TCG Architecture ... 5
4.1 Fundamental Trusted Platform Features ... 5
4.1.1 Protected Capabilities ... 5
4.1.2 Attestation ... 5
4.1.3 Integrity Measurement, Storage and Reporting ... 6
4.2 The Trusted Platform ... 6
4.2.1 Trusted Platform Building Blocks ... 7
4.2.2 The Trust Boundary ... 7
4.2.3 Transitive Trust ... 7
4.2.4 Integrity Measurement ... 8
4.2.5 Integrity Reporting ... 9
4.2.6 TPM as an Endpoint of Communication ... 15
4.2.7 Protected Storage ... 16
4.3 Trusted Platform Module (TPM) Components ... 19
4.3.1 Discrete Components ... 19
4.3.2 Communications Interface ... 20
4.3.3 Tamper-Protected Packaging ... 21
4.4 Privacy Considerations ... 21
4.4.1 What Does TCG Protect? ... 21
4.4.2 How Does TCG Protect It? ... 21
4.5 TCG Execution Model ... 21
4.5.1 TPM Operational States ... 22
4.5.2 Platform Operation ... 24
4.5.3 Interfacing with TPM and Software Services ... 25
4.6 TCG Programmatic Interfaces ... 34
4.6.1 Naming Conventions ... 34
4.6.2 Command Ordinals & Serialization ... 34
4.6.3 Summary of TCG Commands and Interfaces ... 35
5. TCG Model for Security Evaluation ... 41
5.1 The Context for Evaluation ... 41
5.2 Goal of Evaluation ... 42
5.3 The Evaluation Process ... 43
5.3.1 Inputs to Evaluation ... 43
5.3.2 Evaluation Results ... 45
5.4 Certification ... 45
5.4.1 Certified Products List ... 45
5.4.2 Where Does Certification Authority Originate? ... 45
5.5 Accreditation ... 46
5.5.1 Protection Profile as Security Policy ... 46
5.5.2 Site-Specific Security Policy ... 46
5.5.3 Accreditation and Attestation ... 46
5.6 TCG Specification Conformance ... 46
6. Manufacturing & Support Implications of TPM ... 47
6.1 Tamper-resistant Packaging ... 47



IPR2025-01572
Apple EX1048 Page 5

6.2 Field-Upgrade ..... 47
6.3 International Import/Export of Cryptography ..... 47
6.4 Key Management Infrastructure ..... 47
7. Glossary ..... 48

# Table of Figures

Figure 4:a - Reference PC Platform Containing a TCG Trusted Platform Module (TPM). ..... 5
Figure 4:b – Bold indicates part of Trusted Building Block components of a trusted platform ..... 7
Figure 4:c – Transitive trust applied to system boot from a static root of trust. ..... 8
Figure 4:d - Attestation Protocol and Message Exchange ..... 9
Figure 4:e - Diagram of Credentials and their Relationships. ..... 14
Figure 4:f – Root of Trust for Storage (RTS) Architecture ..... 17
Figure 4:g – TPM Component Architecture ..... 19
Figure 4:h – Suggested default operational mode set by platform manufacturer. ..... 22
Figure 4:i – TCG Software Layering. ..... 26
Figure 4:j – Application Interaction Scénarios. ..... 28
Figure 4:k - Command Validation Sessions and Endpoints ..... 30
Figure 4:l - OIAP Sequence ..... 32
Figure 4:m - Object Creation Using ADIP ..... 33
Figure 4:n - Updating Child Authorization Data using ADCP ..... 33
Figure 4:o - TPM Commands Summary ..... 37
Figure 4:p - TDDL Functions Summary ..... 37
Figure 4:q - TCS Functions Summary ..... 38
Figure 4:r - TSP Objects and Methods Summary ..... 40
Figure 5:a - ISO 15408 General Security Context ..... 41
Figure 5:b – ISO 15408 Assurance Context for Evaluation ..... 42
Figure 5:c - ISO15408 Evaluation Context with Process Flow. ..... 43



IPR2025-01572
Apple EX1048 Page 6

# 1. Scope and Audience

This Architectural Overview provides an introduction to TCG goals and architecture. It defines anticipated scenarios for use of TPM enabled platforms, compliance procedures and anticipated implications on manufacturing and support processes.

Anyone looking for an overview of TCG specification and its implications should read this document.

This document does NOT contain normative text.



IPR2025-01572
Apple EX1048 Page 7

# 2. What is TCG?

## 2.1 History

The Trusted Computing Group (TCG) is a not-for-profit industry-standards organization with the aim of enhancing the security of the computing environment in disparate computer platforms. TCG was formed in spring 2003 and has adopted the specifications developed by the Trusted Computing Platform Alliance (TCPA). The distinguishing feature of TCG technology is arguably the incorporation of “roots of trust” into computer platforms.

## 2.2 Mission

Through the collaboration of platform, software, and technology vendors develop a specification that delivers an enhanced HW and OS based trusted computing platform that enhances customers' trusted domains.

## 2.3 Goals

The Trusted Computing Group (TCG) will publish specifications defining architectures, functions and interfaces that provide a baseline for a wide variety of computing platform implementations. Additionally, the TCG will publish specifications describing specific platform implementations such as the Personal Computer, Personal Digital Assistants (PDA), Cellular telephones and other computing equipment.

Platforms based on the TCG specifications are expected to meet functional and reliability standards that allow increased assurance of trust. The TCG will publish evaluation criteria and platform specific profiles that may be used as a common yard stick for evaluating devices incorporating TCG technology.

Achieving improved trust also requires operational integrity of maintenance processes after deployment. The TCG will recommend practices and procedures for maintaining trust in deployed platforms.

# 3. TCG Usage Scenarios

## 3.1 Risk Management

The goal of risk management is to minimize the risk to corporate and personal assets due to malicious and accidental loss or exposure. Risk management processes help assess and mitigate risk. An element of risk management is vulnerability assessment. Asset owners seek to understand techniques employed to protect their assets and identify vulnerabilities associated with the protection mechanisms.

TCG technologies such as Protected Storage can be applied to reduce the risk to information assets Protected storage can be used for securing public, private and symmetric keys that may be especially threatened since access to these represents access to a broader class of information assets. Since protected storage is based on mechanisms that are implemented in an isolated sub-system, the keys can be made less vulnerable to attack.

To minimize risk, information managers naturally seek to protect information assets. This can be accomplished with cryptographic hashing to detect loss of integrity; public and secret key encryption to prevent unauthorized disclosure and digital signing to authenticate transmitted information. The TCG Protected Storage mechanisms rooted in hardware can then be used to protect keys, secrets and hash values. The vulnerability factor (used when computing Loss Expectancy) will decrease when information assets are protected in this way.

TCG systems follow ISO-15408 guidelines for evaluation and certification allowing information managers additional assurance that TCG mechanisms are implemented properly.

## 3.2 Asset Management

Asset managers seek to prevent theft and unauthorized use of computing assets. Asset tracking can be an effective tool in achieving asset management objectives. TCG-defined Trusted Platform Modules (TPM) are manufactured such that ownership of a platform can be asserted by asset managers while allowing users ability to perform job functions.

Under owner control the TPM can be used to create and protect an identity for the system that is not intended to be physically removed or replaced. Asset databases may use this identity to more reliably associate platform asset information.

If an asset is stolen, the thief cannot gain access to information assets, hence may not profit from the consumption or brokering of stolen information.

## 3.3 E-commerce

Customer loyalty and vendor trust are important ingredients in electronic commerce interactions. Vendors build trust, in part, when transactions go smoothly and customer preferences are accurately reflected. Repeat business and loyalty is more likely when customers are able to recall the context of prior positive on-line transactions with vendors.

TCG technology gives platforms the ability to define an e-commerce context in which customer and vendor may establish a relationship based on information exchange. Customers are able to control preferences that may be important to both customer and vendor. If the customer desires, a vendor can identify repeat customers and trust customer-managed preferences; by verifying the relationship context dynamically.

The Trusted Platform Module (TPM) can report platform configuration information, that can be used to define the customer relationship context. The report is cryptographically verifiable enabling both parties the opportunity to be assured that the e-commerce transaction occurs in the context of the previously established relationship.

## 3.4 Security Monitoring and Emergency Response

IT managers expend a great deal of their time responding to virus attacks and threats. Emergency response teams must react quickly to isolate and inoculate vulnerable systems. Often they are required to



IPR2025-01572
Apple EX1048 Page 9

scan the configurations and settings of all the enterprise connected systems to determine which systems need to be updated..

A TPM can be used to ensure that each computer will report its configuration parameters in a trustworthy manner. Platform boot processes are augmented to allow the TPM to measure each of the components in the system (both hardware and software) and securely store the results of the measurements in Platform Configuration Registers (PCR) within the TPM. Emergency response personnel can use these measurements to determine which computers are vulnerable to virus attacks..

IT managers may install system processes that use the PCR values in a TPM to identify unsafe configurations at system boot thereby preventing inadvertent network connection while in an unsafe mode.

# 4. TCG Architecture

For illustrative purposes, this document defines generic, system architectures – also called “reference architectures”. For example, Figure 4:a shows a reference architecture for a Personal Computer (PC). The reference architecture diagrams provide a backdrop for discussion of various TCG concepts.

TCG reference architectures for Personal Computers (PC), Personal Digital Assistants (PDA), cellular telephones and other types of platforms are introduced as illustrative aids. **The reference architecture should not be construed as a bias of the TCG towards any particular platform architecture or implementation**.

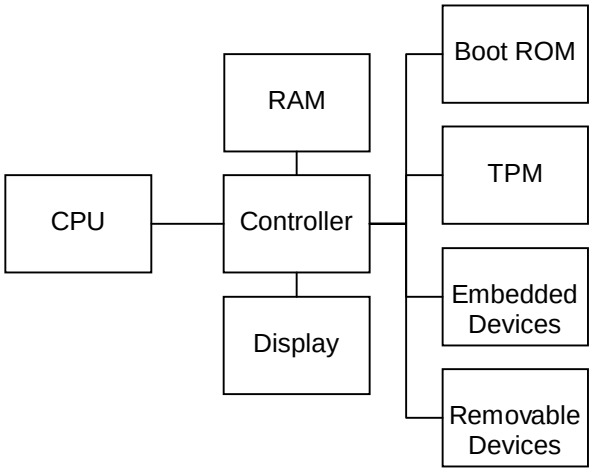



Figure 4:a - Reference PC Platform Containing a TCG Trusted Platform Module (TPM).

## 4.1 Fundamental Trusted Platform Features

Trust is the expectation that a device will behave in a particular manner for a specific purpose. A trusted platform should provide at least three basic features: *protected capabilities*, integrity *measurement* and *integrity reporting.*

### 4.1.1 Protected Capabilities

Protected capabilities are a set of commands with exclusive permission to access shielded locations. Shielded locations are places (memory, register, etc.) where it is safe to operate on sensitive data; data locations that can be accessed only by protected capabilities.

The TPM implements protected capabilities and shielded-locations used to protect and report integrity measurements (called Platform Configuration Registers: PCRs).

The TPM also stores cryptographic keys used to authenticate reported measurements.

TPM protected capabilities can include additional security functionality such as cryptographic key management, random number generation, sealing data to system state and others as determined necessary by TCG members.

### 4.1.2 Attestation

Attestation is the process of vouching for the accuracy of information. External entities can attest to shielded locations, protected capabilities, and Roots of Trust. A platform can attest to its description of platform characteristics that affect the integrity (trustworthiness) of a platform. All forms of attestation require reliable evidence of the attesting entity.

Attestation can be understood along several dimensions, attestation by the TPM, attestation to the platform, attestation of the platform and authentication of the platform.



IPR2025-01572
Apple EX1048 Page 11

*Attestation by the TPM* is an operation that provides proof of data known to the TPM. This is done by digitally signing specific internal TPM data using an attestation identity key (AIK). The acceptance and validity of both the integrity measurements and the AIK itself are determined by a verifier. The AIK is obtained using either the Privacy CA or via a trusted attestation protocol.

*Attestation to the platform* is an operation that provides proof that a platform can be trusted to report integrity measurements; performed using the set or subset of the credentials associated with the platform; used to issue an AIK credential.

*Attestation of the platform* is an operation that provides proof of a set of the platform's integrity measurements. This is done by digitally signing a set of PCRs using an AIK in the TPM.

*Authentication of the platform* provides evidence of a claimed platform identity. The claimed identity may or may not be related to a user or any actions performed by the user. Platform Authentication is performed using any non-migratable signing key. Certified keys (i.e. signed by an AIK) have the added semantic of being attestable. Since there are an unlimited number of non-migratable keys associated with the TPM, there are an unlimited number of identities that can be authenticated.

### 4.1.3 Integrity Measurement, Logging and Reporting

*Integrity measurement* is the process of obtaining metrics of platform characteristics that affect the integrity (trustworthiness) of a platform and putting digests of those metrics in PCRs.

The starting point of measurement is called the root of trust for measurement. A static root of trust for measurement begins measuring from a well-known starting state such as a power on self-test. A dynamic root of trust for measurement transitions from an un-trusted state to one that is trusted.

An optional intermediate step between integrity measurement and integrity reporting is *integrity logging*, which stores integrity metrics in a log for later use. Logging is recommended. Otherwise integrity measurements might need to be repeated in order to interpret PCR values.

*Integrity reporting* is the process of attesting to integrity measurements recorded in PCRs.

The philosophy of integrity measurement, logging and reporting is that a platform may be permitted to enter any state possible including undesirable or insecure states, but that it may not be permitted to lie about states that it was or was not in. An independent process may evaluate the integrity state(s) and determine an appropriate response.

## 4.2 The Trusted Platform

In TCG systems *roots of trust* are components that must be *trusted* because misbehavior might not be detected. A complete set of Roots of Trust has at least the minimum functionality necessary to describe the platform characteristics that affect the trustworthiness of the platform.

There are commonly three *Roots of Trust* in a trusted platform; a root of trust for measurement (RTM), root of trust for storage (RTS) and root of trust for reporting (RTR). The RTM is a computing engine capable of making inherently reliable integrity measurements. Typically the normal platform computing engine, controlled by the core root of trust for measurement (CRTM). The CRTM is the instructions executed by the platform when it acts as the RTM. The RTM is also the root of the chain of *transitive trust* (see 4.2.3). The RTS is a computing engine capable of maintaining an accurate summary of values of integrity digests and the sequence of digests. The RTR is a computing engine capable of reliably reporting information held by the RTS.

Each root is trusted to function correctly without external oversight. Trusting "roots of trust" may be achieved through a variety of ways but is anticipated to include technical evaluation by competent experts.



IPR2025-01572
Apple EX1048 Page 12

### 4.2.1 Trusted Platform Building Blocks

Trusted Building Blocks (TBB) are the parts of the Roots of Trust that do not have shielded locations or protected capabilities. Normally these include just the instructions for the RTM and TPM initialization functions (reset, etc.). Typically they are platform-specific. One example of a TBB is the combination of the CRTM, connection of the CRTM storage to a motherboard, the connection of the TPM to a motherboard, and mechanisms for determining Physical Presence (see Figure 4:b).

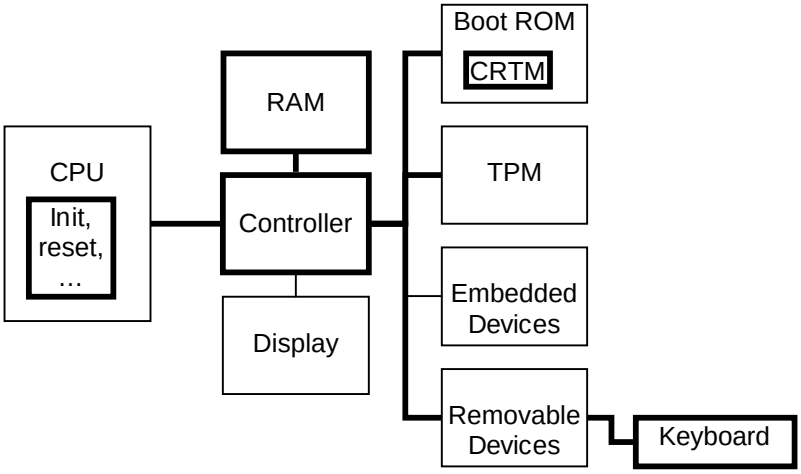



Figure 4:b – Bold indicates part of Trusted Building Block components of a trusted platform

The TBB is trusted, meaning it is expected to behave in a way that doesn't compromise the goals of trusted platforms.

### 4.2.2 The Trust Boundary

The combination of TBB and Roots of Trust form a trust boundary where measurement, storage and reporting can be accomplished for a minimal configuration. More complex systems may require measurements be taken by other (optional) ROM code besides the CRTM. For this to occur trust in other ROM code must be established. This is done by measuring the ROM code prior to transferring execution control. The TBB should be established such that devices containing other measurement code do not inadvertently extend the TBB boundary where trustworthiness of the linkages has not been previously established.

### 4.2.3 Transitive Trust

Transitive trust also known as "Inductive Trust", is a process where the Root of Trust gives a trustworthy description of a second group of functions. Based on this description, an interested entity can determine the trust it is to place in this second group of functions. If the interested entity determines that the trust level of the second group of functions is acceptable, the trust boundary is extended from the Root of Trust to include the second group of functions. In this case, the process can be iterated. The second group of functions can give a trustworthy description of the third group of functions, etc. Transitive trust is used to provide a trustworthy description of platform characteristics, and also to prove that non-migratable keys are non-migratable

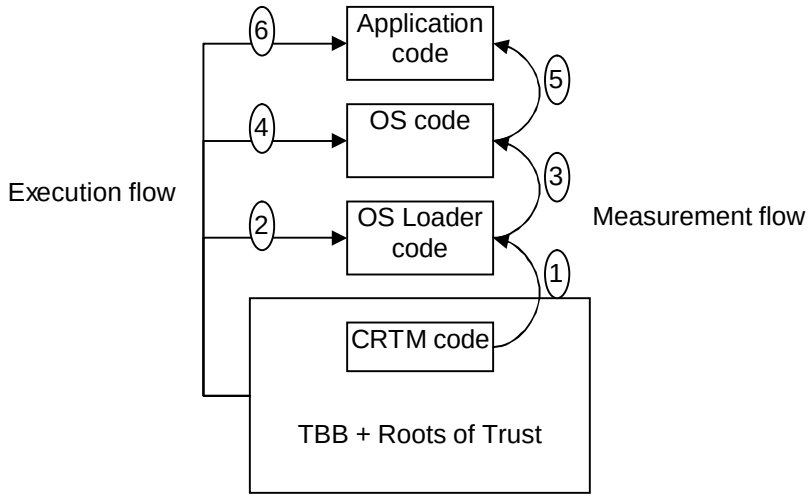



Figure 4:c – Transitive trust applied to system boot from a static root of trust.

In Figure 4:c transitive trust is applied to a system booting from a static root of trust and the trust boundary is extended to include code that didn't natively reside within the roots of trust. In each extension of the trust boundary, the target code is first measured before execution control is transferred.

### 4.2.4 Integrity Measurement

A measurement kernel generates *measurement events*. A measurement event consists of two classes of data; 1) measured values - a representation of embedded data or program code and 2) measurement digests - a hash[1] of those values. Data are scanned by the measurement kernel which generates a message digest. Digests are a snapshot of the machines operational state. The two data elements (measured values and measurement digest) are stored separately. The measurement digest is stored in the TPM using RTR and RTS functionality. The measured values[2] may be stored virtually anywhere at the discretion of the measurement kernel. In fact, it may not be stored at all, but re-computed whenever the serialized representation is needed.

Measurement data describe properties and characteristics of the measured component. It is the responsibility of the measurement kernel implementer to understand the syntax and semantics of measured fields in sufficient detail to produce an encoding suitable for measurement event consumers. Implementers play a role in determining how event data may be partitioned. The platform specific specifications contain additional insight in specifying the *platform configuration*, its representation and anticipated measurement consumers.

The *Stored Measurement Log* (SML) contains sequences of related measured values. Each sequence shares a common measurement digest. Measured values are appended to the common measurement digest and re-hashed. This is more commonly referred to as *extending* the digest. Extending ensures related measured values will not be ignored and order of operations is preserved.

The TPM contains a set of registers, called *Platform Configuration Registers* (PCR) containing measurement digests. Algebraically, updates to a PCR follows as: PCR[n] ß SHA-1 ( PCR[n] + measured data). PCR values are temporal and are reset at system reboot.

Verification of measurement events requires recreation of the measurement digest and a simple compare of digest values (using the PCR value as one of the comparators). TCG does not define data encoding rules for SML contents but recommends following appropriate standards such as Extensible Markup Language (XML) to ensure broad accessibility. Nevertheless, different platforms may require different

[1] E.g. a Sha-1 cryptographic hash.
[2] More commonly known as the Stored Measurement Log (SML)

representation, hence the Platform Specific Specifications (e.g., the PC-Specific Platform Specification) may define other encoding rules.

The SML can become very large. Therefore it does not reside in the TPM. The SML does not need the protection afforded by the TPM as attacks against the SML would be detected. However, SML is still subject to denial of service attacks. Implementers should take steps to replicate or regenerate the log.

### 4.2.5 Integrity Reporting

The Root of Trust for Reporting (RTR) has two functions, to expose shielded-locations for storage of integrity measurements. A second objective is to *attest* to the authenticity of stored value based on trusted platform identities. PCRs may be implemented in volatile or non-volatile storage. PCRs must be protected from software attack. Steps to prevent physical tampering should be taken into consideration. Integrity reports are digitally signed to authenticate PCR values using Attestation Identity Keys (AIK). A nonce[3] is included with the signed PCRs to prevent replay.

The TPM generates and manages AIK keys. TPMs can have many AIK keys. A different AIK is used to protect privacy when the platform owner is concerned about the consequences of collusion.

TPMs can be shipped with an embedded key called the Endorsement Key (EK). The EK is used in a process for the issuance of AIK credentials and to establish a platform owner. The platform owner can create a storage root key. The storage root key in turn is used to wrap other TPM keys. See sections: 4.2.5.2, 4.2.7, and 4.4 for additional details regarding EK.

#### *4.2.5.1 Integrity Reporting Protocol*

Integrity reporting may be used to determine a platform's current configuration. A protocol for reporting integrity measurements is illustrated in Figure 4:d

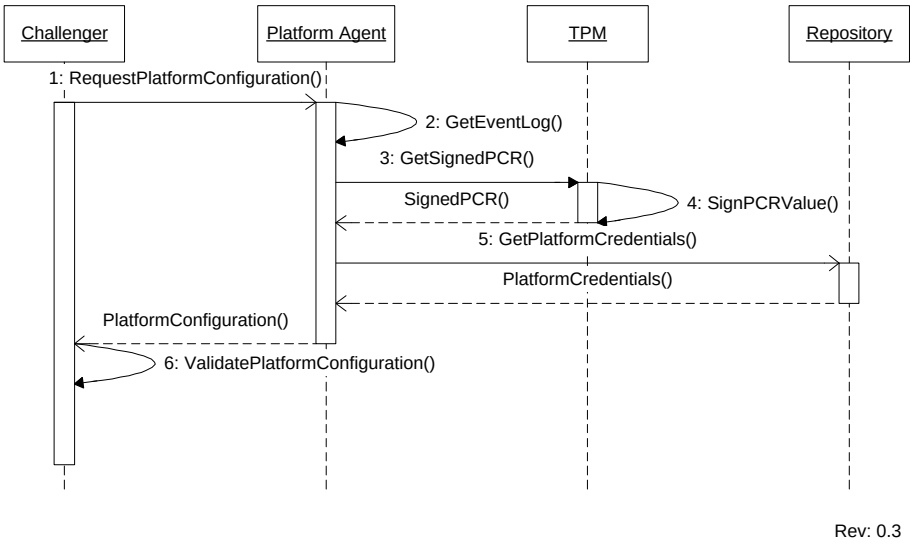



Figure 4:d - Attestation Protocol and Message Exchange

Attestation protocol consists of several steps:

1. A Challenger requests one or more PCR values from a platform.
2. An agent on the platform containing a TPM, collects SML entries.
3. The Platform Agent receives PCR values from the TPM.

[3] A nonce is an unpredictable random value used during a cryptographic operation to prevent replay attack.

4. The TPM signs PCR values using an AIK.
5. The Platform Agent collects credentials that vouch for the TPM. The signed PCR value, SML entries and Credentials are returned to the Challenger.
6. The Challenger verifies the request. The measurement digest is computed and compared with PCR value. The platform credentials are evaluated and signatures checked.

The protocol is independent of transport or delivery mechanism. It is anticipated that existing communications, messaging and remote procedure call infrastructures can be leveraged to transport attestation messages.

#### *4.2.5.2 Identity and Privacy*

One objective of attestation is to allow the Challenger to determine that *some* TPM has signed a message. It may also be used to determine *"which"* TPM signed the message. A Privacy CA may be employed to issue AIK credentials that vouch for the trustworthiness of a platform without disclosing EK unique values to a Challenger. The TPM enrolls AIK public keys with a Privacy CA. The Privacy CA may then distribute a credential certifying the AIK. Enrollment with a Privacy CA requires the TPM to prove AIK keys are exclusively bound to the TPM. The platform accomplishes this by decrypting the AIK credential using the EK private key in the TPM. Only the TPM with the EK private key will be able to perform the decryption.

The Privacy CA is trusted not to reveal sensitive information. This includes the public EK or PII derived from it. It is also trusted not to misrepresent the trust properties of platforms for which AIK credentials are issued.

A TPM can be configured to require owner authorization before participating in AIK credential issuance protocols. A TPM can further be disabled or deactivated to further control TPM use.

#### *4.2.5.3 Attestation of the Platform*

Attestation of the platform evaluates evidence (not residing in the platform) that is used to determine if an AIK can be trusted to authenticate a platform. Credentials issued by TPM and platform manufacturers are needed to validate trust assertions that may be associated with a particular AIK.

#### *4.2.5.4 TCG Credentials*

TCG defines five types of credentials. Each type is intended to provide only the information necessary to perform a specific operation.

Credential formats are expressed in ASN.1 notation and are expected to be able to leverage some elements of public key infrastructure.

Credential types include:

- Endorsement or EK credential
- Conformance credential
- Platform credential
- Validation credential
- Identity or AIK credential

##### *4.2.5.4.1 Endorsement Credential*

This credential is issued by whoever generates the EK. The EK is generated as part of the manufacturing process. It is expected that the TPM vendor will generate the EK. However, the EK should be generated at any point prior to final delivery to the end customer for the manufacturer to claim that the EK was properly created and embedded within a valid TPM.

If the EK key pair is generated after delivery of the platform to a customer, the conditions in which the key was created may impact the endorsement that can be provided.

The Endorsement Credential contains the following information:

- TPM Manufacturer Name
- TPM Part Model Number
- TPM Version or Stepping
- EK Public Key

The EK public key, though public, is privacy-sensitive due to the fact that it uniquely identifies the TPM and by extension the platform. TCG anticipates one *EK credential* is needed per TPM instance.

#### *4.2.5.4.2 Conformance Credentials*

These are issued by anyone with sufficient credibility to evaluate a TPM or platforms containing TPMs. The evaluation could be performed by the platform manufacturer, vendor or independent entity. The conformance credential indicates the evaluator accepts that the TBB design and implementation in accordance with established evaluation guidelines. By signing the credential, the evaluator attests to the evaluation result, the details of which may be available for inspection. TCG facilitates evaluations by defining meaningful evaluation criteria and guidelines.

Multiple Conformance Credentials may be issued for a single platform; one for the TPM and others for disparate TBB components (depicted as B in Figure 4:e).

The Conformance Credential(s) may contain the following information:

- Evaluator Name
- Platform Manufacturer Name
- Platform Model Number
- Platform Version (if applicable)
- TPM Manufacturer Name
- TPM Model Number
- TPM Version or Stepping

Conformance Credentials do not contain information that uniquely identifies any particular platform.

TCG envisages several Conformance Credentials may exist per platform model but that only one set of credentials are needed for multiple platforms of the same make and model.

#### *4.2.5.4.3 Platform Credential*

This credential is issued by the platform manufacturer, vendor or anyone with sufficient credibility. It identifies the platform's manufacturer and describes platform properties. It also references the platform Endorsement Credential associated with the TPM and related Conformance Credentials (See A in Figure 4:e). Credential references consist of a message digest of the referred credential. Platform Credentials could be considered privacy-sensitive: the credential is associated with a specific platform, as opposed to a class of such platforms.

The Platform Credential contains the following information:

- Platform Manufacturer Name
- Platform Model Number
- Platform Version (if applicable)
- Endorsement Credential
- Conformance Credential(s)

The Platform Credential provides evidence that the platform contains a TPM as described by the Endorsement Credential. TCG envisages there will be one *Platform Credential* for each platform instance.

#### *4.2.5.4.4 Validation Credential*

TCG envisages that manufacturers of measurable components, hardware or software, will provide reference measurement values Reference measurements are digests of measured components taken during manufacturing when the component is believed to be in proper working order. Typically, this occurs after functional testing. Not all components will have validation credentials created. Only components that pose a threat to security should be vetted for back-doors.

Clean-room measurements may be taken at anytime with the caveat that consumers of expected digests trust the clean-room operators' claims. Reference measurements can be compared to actual (runtime) measurements enabling detection of changes. The signed documents describing component structures with expected digests are called *Validation Credentials*.

Examples of components that might have Validation Credentials include:

- Video adapters
- Disk storage adapters
- Memory controllers
- Communications controllers / Network adapters
- Processors
- Keyboard and mouse
- Software

Validation credentials are issued by a *validation entity*. Anyone willing and able to take measurements and attest to measured values may be regarded as a validation entity. Typically, the component manufacturer is best able to produce expected values. Any part of the components description may be fodder for trust decisions. However, candidate components for validation credentials will likely be those that present a threat to security..

Component descriptions are expected to include at least the following elements:

- Validation Entity Name
- Component Manufacturer Name
- Component Model Number
- Component Version or Stepping
- Measurement Value(s)

Component descriptions may contain:

- Component Capabilities (immutable)

There may be one credential issued per model, series of components, or individually as determined by the uniqueness of information being signed. Validation credentials are distributed and published by any means deemed suitable by those performing validation. It is hoped publications will be provided in electronic form that is readily consumable by automated tools.

Agents wishing to prove accuracy of Event data may compare PCR values to Validation Credential measurement values.

##### *4.2.5.4.4.1 Component Updates and Field-Upgrade*

Components having validation credentials may require bug fixing, updates or upgrades that would nullify expected results contained in original validation credentials. TCG believes the processes governing VC creation and component updates are vendor specific. The platform-specific specifications will address component-VC consistency issues.



IPR2025-01572
Apple EX1048 Page 18

#### *4.2.5.4.5 Attestation Identity Credential*

The AIK credential identifies the AIK private key used to sign PCR values. It contains the AIK public key and optionally any other information deemed useful by the issuer. Attestation Credentials are issued by a service that is trusted to verify the various credentials and preserve privacy policies of the client..

By issuing the Attestation Identity Credential, the signer attests to TPM authenticity by proving facts about the TPM. Goals of the proof are that the TPM owns the AIK and the AIK is tied to valid Endorsement, Platform and Conformance credentials. The trusted party further guarantees that it will abide by privacy expectations of the client. Expectations may include protection of personally identifiable information that may have been disclosed as part of enrollment processes.

Attestation Identity Credentials reference other credentials as follows:

- The Attestation Identity credential contains a reference to the TPM manufacturer and model depicted as C in Figure 4:e, but not the privacy-sensitive EK.
- The Attestation Identity credential also contains a reference to the platform manufacturer and model depicted as D in Figure 4:e. Note, however, this reference is not to the platform credential itself, rather it is a reference to the information contained within the platform credential that is not privacy-sensitive.
- Finally, the Attestation Identity credential contains a pointer to where a challenger can find the TPM and platform's conformance documentation as depicted as E in Figure 4:e.

A challenger could use this information, along with other information in the credential to trust the platform via Attestation protocol.



IPR2025-01572
Apple EX1048 Page 19

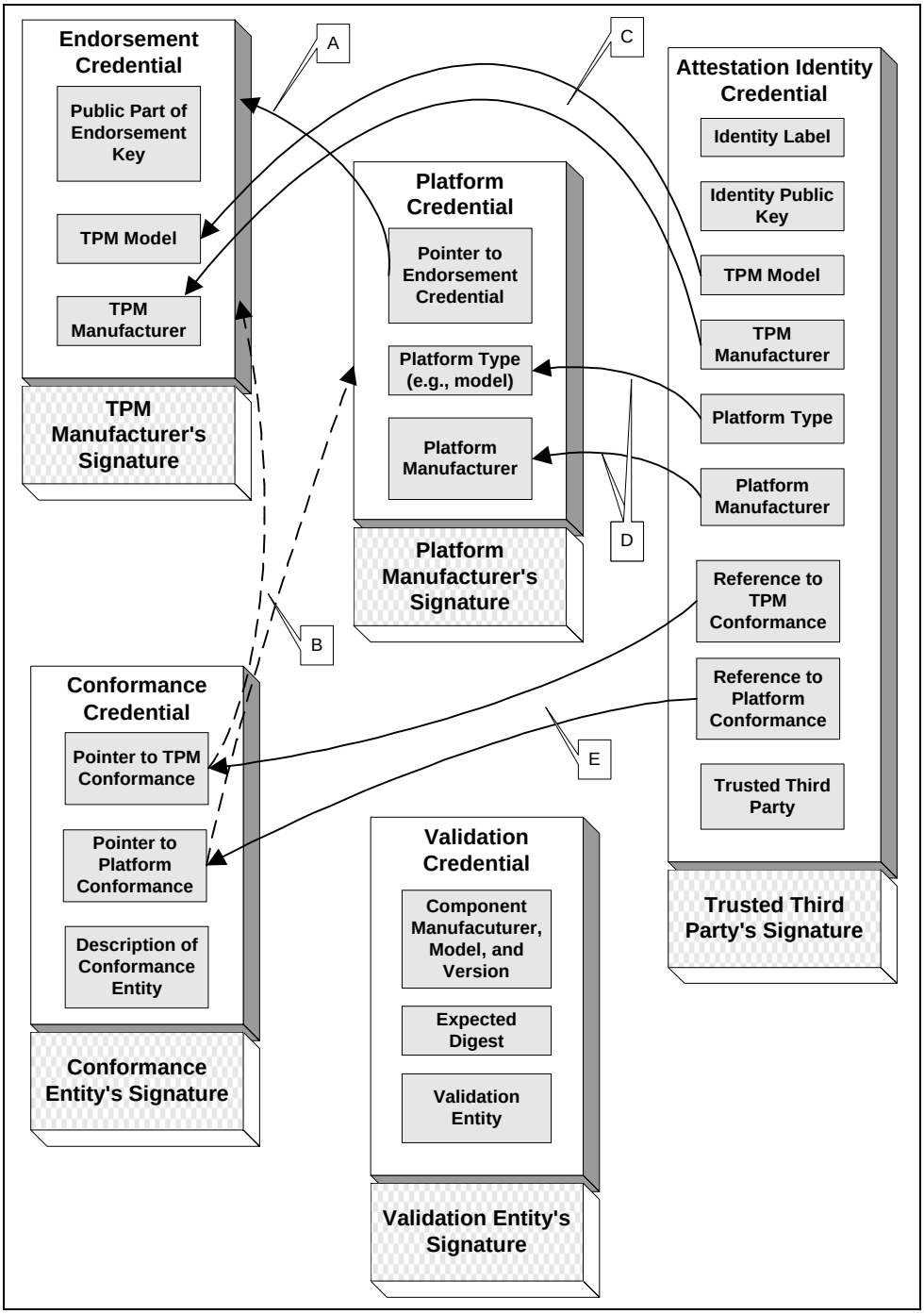



Figure 4:e - Diagram of Credentials and their Relationships.

#### *4.2.5.4.6 Credentials Trust and Privacy*

Within each description of the above credentials, a statement about the privacy-sensitivity was made. While each of the above credentials is needed to provide trust in the platform, the TCG specification dictates that platform manufacturers must provide protection of these credentials for the purpose of maintaining the user's privacy.

There are several methods for distributing attestation credentials. They include:

- Distribution with the platform – on CD or magnetic media.
- Website download – URL to manufacturer web page.
- 3rd party service – Such as a search engine or directory service.

Regardless of the mechanism for obtaining the credentials, distributors of credentials are expected to provide mechanisms to meet privacy expectations of the platform owner.

### 4.2.6 TPM as an Endpoint of Communication

Designers of secure distributed systems, when considering exchange of information between systems, must identify the endpoints of communication. The composition and makeup of the endpoint is as important to the overall security of the system as is the communications protocol. TCG designers assert endpoints are minimally comprised of asymmetric keys, key storage and processing that protects protocol data items.

Classic message exchange based on asymmetric cryptography suggests that messages intended for one and only one individual can be encrypted using a public key. Furthermore, the message can be protected from tampering by signing with the private key. Keys are communication endpoints and improperly managed keys can result in loss of security. Additionally, improperly configured endpoints may also result in loss of security. The TPM aids in improving security by providing both key management and configuration management features (E.G. Protected Storage, Measurement and Reporting).

These features can be combined to "seal" keys and platform configuration making endpoint definition stronger.

TCG defines four classes of protected message exchange; Binding, Signing, Sealed-Binding (A.K.A. Sealing) and Sealed-Signing.

#### *4.2.6.1 Binding*

*Binding* is the traditional operation of encrypting a message using a public key. That is, the sender uses the public key of the intended recipient to encrypt the message. The message is only recoverable by decryption using the recipient's private key. When the private key is managed by the TPM as a non-migratable key only the TPM that created the key may use it. Hence, a message encrypted with the public key, "bound" to a particular instance of a TPM.

It is possible to create migratable private keys that are transferable between multiple TPM devices. As such, binding has no special significance beyond encryption.

#### *4.2.6.2 Signing*

Signing also in the traditional sense, associates the integrity of a message with the key used to generate the signature. The TPM tags some managed keys as signing only keys, meaning these keys are only used to compute a hash of the signed data and encrypt the hash. Hence, they cannot be misconstrued as encryption keys.

#### *4.2.6.3 Sealing*

*Sealing* takes binding one step further. Sealed messages are bound to a set of platform metrics specified by the message sender. Platform metrics specify platform configuration state that must exist before decryption will be allowed. Sealing associates the encrypted message (actually the symmetric key used to encrypt the message) with a set of PCR register values and a non-migratable asymmetric key.

A sealed message is created by selecting a range of PCR register values and asymmetrically encrypting the PCR values plus the symmetric key used to encrypt the message. The TPM with the asymmetric decryption key may only decrypt the symmetric key when the platform configuration matches the PCR register values specified by the sender.

Sealing is a powerful feature of the TPM. It provides assurance that a protected messages are only recoverable when the platform is functioning in a very specific known configuration.

#### *4.2.6.4 Sealed-Signing*

Signing operations can also be linked to PCR registers as a way of increasing the assurance that the platform that signed the message meets a specific configuration requirement. The verifier mandates that a signature must include a particular set of PCR registers. The signer, during the signing operation, collects the values for the specified PCR registers and includes them in the message, and as part of the computation of the signed message digest. The verifier can then inspect the PCR values supplied in the signed message, which is equivalent to inspecting the signing platform's configuration at the time the signature was generated.

### 4.2.7 Protected Storage

The Root of Trust for Storage (RTS) protects keys and data entrusted to the TPM. The RTS manages a small amount of volatile memory where keys are held while performing signing and decryption operations. (See Key Slots in Figure 4:f). Inactive keys may be encrypted and moved off-chip to make room for other more active keys. Management of the key slot cache is performed external to the TPM by a Key Cache Manager (KCM). The KCM interfaces with a storage device where inactive keys may be stored indefinitely. The RTS doubles as a general purpose protected storage service allowing opaque data also to be stored.

The RTS is optimized to store small objects roughly the size of an asymmetric key minus overhead (e.g. ~210 byte payload). A variety of object types can be stored, such as symmetric and asymmetric keys, pass-phrases, cookies, authentication results and opaque data.

There are three key types that are not opaque to the TPM. AIK keys, Signing keys and Storage keys. Key types will be discussed in more detail later. Two keys are embedded in the TPM, see Figure 4:f, the Storage Root Key (SRK) and the Endorsement Key (EK) - previously discussed. These keys cannot be removed from the TPM. However, a new SRK may be created as part of creating a new platform owner, This has the side-effect of leaving encrypted all data objects controlled by the previous SRK.

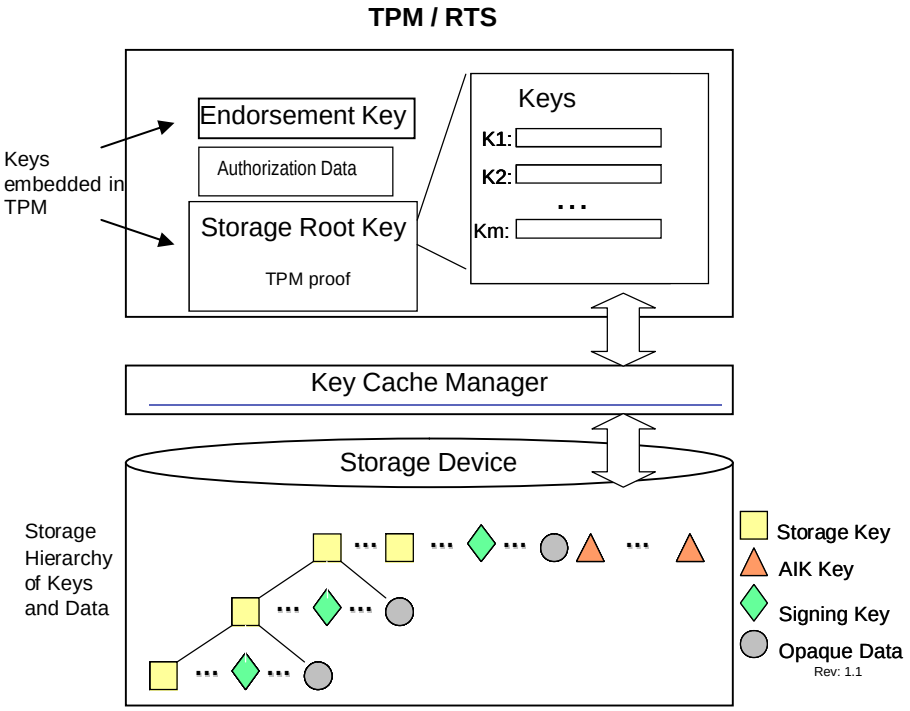



Figure 4:f – Root of Trust for Storage (RTS) Architecture

The SRK is generated by the TPM and the SRK pass phrase is encrypted using the EK when a TPM owner is established. The SRK is used to protect other keys that may be stored externally to the TPM.

#### *4.2.7.1 Key Attributes*

All keys managed by the RTS have an attribute designation of *migratable* or *non-migratable*. The key attribute determines whether a key may be transferred from one TPM to another. Attribute value is established at the time the key is created and cannot be changed. Semantically, a non-migratable key is permanently associated with a specific TPM instance. Migration of a non-migratable key would have the effect of allowing one platform to masquerade as another. An AIK key is a prime example of a key that should never be migrated. Hence, AIK keys are fixed as non-migratable.

Migratable keys may be exchanged between TPM devices. This enables the key pair to follow the user around irregardless of the device he/she uses. Messages exchanged between individuals remain accessible even though the computing platform changes.

Key attributes may not be applied to opaque data. To extend the non-migratable property to opaque data, the data should be stored with the RTS using a non-migratable storage key. As long as the opaque object is under the control of the TPM, it cannot be decrypted elsewhere. However, once data are decrypted external to a TPM, it can obviously be migrated to any system.

#### *4.2.7.2 Key Types*

TCG defines 7 key types. Each type carries with it a set of restrictions that limits its use. TCG keys can be classified broadly as either signing or storage keys. They can be further typed as Platform, Identity, Binding, General and Legacy keys. Symmetric keys are classified separately as Authentication keys.

The 7 key types are listed as follows:

- *Signing keys* are asymmetric general purpose keys used to sign application data and messages. Signing keys can be migratable or non-migratable. Migratable keys may be exported / imported between TPM devices. The TPM can sign application data and enforce migration restrictions.

- *Storage keys* are asymmetric general purpose keys used to encrypt data or other keys. Storage keys are used for wrapping keys and data managed externally (see 4.2.7.3),
- Identity Keys (a.k.a. AIK keys) are non-migratable signing keys that are exclusively used to sign data originated by the TPM (such as TPM capabilities and PCR register values).
- *Endorsement Key* (EK) is a non-migratable decryption key for the platform. It is used to decrypt owner authorization data at the time a platform owner is established and to decrypt messages associated with AIK creation. It is never used for encryption or signing.
- *Bind keys* may be used to encrypt small amounts of data (such as a symmetric key) on one platform and decrypt it on another.
- *Legacy Keys* are keys created outside the TPM. They are imported to the TPM after which may be used for signing and encryption operations. They are by definition migratable.
- *Authentication Keys* are symmetric keys used to protect transport sessions involving the TPM.

#### *4.2.7.3 External Storage and Key Cache Management*

The TPM is anticipated to become a low cost commodity component, suitable for consumer class computing platforms. Therefore, the TPM itself is likely to have limited runtime (volatile) and persistent (non-volatile) storage. TCG usage scenarios suggest unlimited storage may be needed. For this reason TPM external storage and a cache manager are defined.

##### *4.2.7.3.1 External Storage*

To allow for virtually unlimited keys and storage areas the RTS packages keys destined for external storage into encrypted key BLOBs. Key blobs are opaque outside the TPM and may be stored on any available storage device (e.g. Flash, Disk, and Network File Server). BLOB structures are *bound* to a particular TPM and may be *sealed* to a particular platform configuration as well.

Blobs are referenced using a cryptographic hash of its contents, by handle or other suitable referencing mechanism. Reference identifiers disambiguate Blobs externally to the KCM or other application program performing the storage functions. Other information including Key Type and Key Attribute are available externally.

##### *4.2.7.3.2 Key Cache Management*

The TPM exposes interfaces that allow external programs the ability to manage the limited storage resources of the TPM. Management function is distinguished from application function by separating the ability to cache keys from the ability to use a key. Key Cache Managers (KCM) will generally only be concerned with caching keys, while applications will be concerned about key usage. A noted exception is storage keys which are used to protect other keys. The KCM will likely control both caching and use of storage keys.

Keys sealed to a particular platform configuration may be loaded even when the platform is outside the intended configuration. This allows flexibility in transitioning the platform between readiness states without impacting its ability to obtain needed keys. Security is maintained because configuration is checked each time it is used, hence loading need not be checked.

In Figure 4:f, the Key Cache Manager is shown as an external program brokering movement of keys between volatile Key Slot memory in the TPM and non-volatile external storage device(s). The KCM tracks available key slots and determines when it is appropriate to expel a key and replace with another. The TPM does not provide proactive notification when Key Slots are depleted or when applications need to use a particular key. As such, application programs may need to inform the KCM when such events occur or the KCM needs to implement a TPM interface layer, through which applications obtain TPM services[4].

The TPM provides interfaces to prepare keys for transitioning between TPM and Storage Device. At no time may the KCM render keys in the clear.

[4] Note: using the KCM as a TPM proxy may have unintended security implications. The KCM may become privy to key usage passphrases thereby making the KCM a central point of attack.

The KCM may implement a model for indexing, storing and retrieving Blobs contained on KCM managed storage devices. This may also include management of pass-phrases necessary for using keys in the TPM.

## 4.3 Trusted Platform Module (TPM) Components

This section describes the logical layout of the TPM and its discrete components. Implementations of TPMs may be done in hardware or software. Implementation details are reserved for Platform Specific Specification documentation. A model that favors hardware interpretations of the TPM specification is presented, but it is sufficiently abstract so as not to exclude software implementations.

The TPM in Figure 4:g shows building blocks for a TPM supporting RTR and RTS functionality. As a building block of a trusted platform TPM components are trusted to work properly without additional oversight. Trust in these components is derived from good engineering practices, manufacturing process and industry review. Evidence of engineering practice and industry review is contained in the Common Criteria (CC) certification results (See 5.3.2 Evaluation Results).

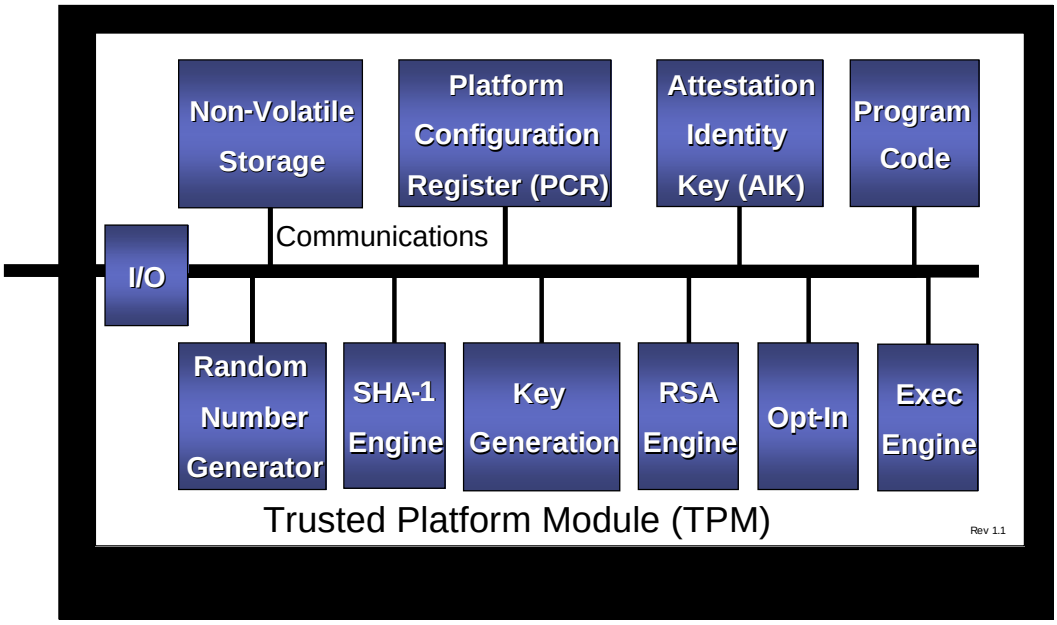



Figure 4:g – TPM Component Architecture

### 4.3.1 Discrete Components

#### *4.3.1.1 Input/Output (I/O)*

The I/O component manages information flow over the communications bus. It performs protocol encoding/decoding suitable for communication over external and internal buses. It routes messages to appropriate components. The I/O component enforces access policies associated with the Opt-In component as well as other TPM functions requiring access control.

#### *4.3.1.2 Non-Volatile Storage*

Non-volatile storage is used to store Endorsement Key (EK), Storage Root Key (SRK), owner authorization data and persistent flags.

Platform Configuration Registers (PCR) can be implemented in either volatile or non-volatile storage. They are reset at system start or whenever the platform loses power. TCG specifies a minimum number



IPR2025-01572
Apple EX1048 Page 25

of registers to implement (16). Registers 0-7 are reserved for TPM use. Registers 8-15 are available for operating system and application use.

#### 4.3.1.3 Attestation Identity Keys(AIK)

Attestation Identity Keys must be persistent, but it is recommended that AIK keys be stored as Blobs in persistent external storage (outside the TPM), rather than stored permanently inside TPM non-volatile storage. TCG hopes TPM implementers will provide ample room for many AIK Blobs to be concurrently loaded into TPM volatile memory as this will speed execution.

#### 4.3.1.4 Program Code

Program code contains firmware for measuring platform devices. Logically, this is the Core Root of Trust for Measurement (CRTM). Ideally, the CRTM is contained in the TPM, but implementation decisions may require it be located in other firmware.

#### 4.3.1.5 Random Number Generator (RNG)

The TPM contains a true random-bit generator used to seed random number generation. The RNG is used for key generation, nonce creation and to strengthen pass phrase entropy.

#### 4.3.1.6 Sha-1 Engine

A Sha-1 message digest engine is used for computing signatures, creating key Blobs and for general purpose use.

#### 4.3.1.7 RSA Key Generation

TCG standardizes the RSA[5] algorithm for use in TPM modules. Its recent release into the public domain combined with its long track record makes it a good candidate for TCG. The RSA key generation engine is use to create *signing keys* and *storage keys*. TCG requires a TPM to support RSA keys up to a 2048-bit modulus, and mandates that certain keys (the SRK and AIKs, for example) must have at least a 2048-bit modulus.

#### 4.3.1.8 RSA Engine

The RSA engine is used for signing with *signing keys*, encryption/decryption with *storage keys*, and decryption with the EK. The TCG committee anticipates TPM modules containing an RSA engine will not be subject to import/export restrictions.

#### 4.3.1.9 Opt-In

The Opt-In component implements TCG policy requiring TPM modules are shipped in the state the customer desires. This ranges from **disabled** and **deactivated** to fully **enabled;** ready for an owner to take possession. The Opt-In mechanism maintains logic and (if necessary) interfaces to determine physical presence state and ensure disabling operations are applied to other TPM components as needed.

#### 4.3.1.10 Execution Engine

The execution engine runs program code. It performs TPM initialization and measurement taking.

### 4.3.2 Communications Interface

The TCG main specification does not specify the communications interfaces or bus architectures. These are considered implementation decisions documented in the Platform Specific Specification(s). TCG does define an interface serialization transformation that can be transported over virtually any bus or interconnect.

[5] RSA was named after its inventors Rivest, Shamir and Adelman.

### 4.3.3 Tamper-Protected Packaging

TCG requires the TPM be physically protected from tampering. This includes physically binding the TPM module (if it were physically a discrete part) to the other physical parts of the platform (e.g. motherboard) such that it cannot be easily disassembled and transferred to other platforms. These mechanisms are intended to *resist* tampering.

Tamper evidence measures are to be employed. Such measures enable *detection* of tampering upon physical inspection.

Software TPM implementations must justify a hardware-equivalent interpretation for tamper-protection. Such an interpretation should realize the desired security properties. Namely, that a particular TPM has cardinality of exactly one and that that TPM is bound to a particular platform.

## 4.4 Privacy Considerations

The TCG technical committee considered the affect each capability may have on privacy. The TCG privacy model generally follows the privacy guiding principles established by the World Wide Web Consortium (W3C) P3P working group[6]. Namely:

- **Notice and Communication** - Service providers should provide timely and effective notices of their information practices, and user agents should provide effective tools for users to access these notices and make decisions based on them.
- **Choice & Control** - Users should be given the ability to make meaningful choices about the collection, use, and disclosure of personal information.
- **Fairness & Integrity** - Users should retain control over their personal information and decide the conditions under which they will share it. Service providers should treat users and their personal information with fairness and integrity. This is essential for protecting privacy and promoting trust.
- **Confidentiality** - Users' personal information should always be protected with reasonable security safeguards in keeping with the sensitivity of the information.

These guidelines are applied in a case-by-case basis by the TCG technical committee to protect the identity of platforms which can be considered personally identifiable information (PII).

### 4.4.1 What Does TCG Protect?

TCG protects the identity of platforms from being trivially discoverable by unknown or unauthorized entities. The platform Endorsement Key (EK), is an asymmetric key pair permanently bound to the platform. The EK can be used to recognize a platform but not identify it.

### 4.4.2 How Does TCG Protect It?

TCG protects identifiers that could be considered PII by disclosing them only when the owner / user allows. It introduces platform identity aliases, known as Attestation Identity Keys (AIK) that may be associated with information relating to a specific use or domain. AIK keys may be recycled by the owner / user.

The TPM itself may be disabled and deactivated. When disabled or deactivated the TPM will not disclose either EK or AIK. The platform owner / operators maintain control over these modes of operation thereby allowing them to implement a privacy policy that balances the ability to attest with privacy constraints.

Additionally the owner can associate the owner's authorization data with the EK, so that releasing the public EK is an authorized command.

## 4.5 TCG Execution Model

The execution model for a TCG TPM begins when the TPM first receives power. This may occur during the manufacturing process prior to end-user delivery of the system. TCG defines a handful of execution

[6] Platform for Privacy Protections (P3P) guiding principles http://www.w3.org/TR/P3P/#guiding_principles

modes or *states* each designed to permit orderly deployment and maintenance of the computing resource while balancing the needs of several stakeholders.

Subsequent to TPM configuration modes, platform configurations can be used to define platform states that can impact execution models for reference monitors and applications.

Finally, a model for interacting with the TPM and supporting services merits exploration.

### 4.5.1 TPM Operational States

There are several mutually-exclusive modes of operation in which TPM behavior may be limited. They are as follows:

- Enabled / Disabled – The TPM may be enabled/disabled multiple times within a boot period. When disabled, the TPM restricts all operations except the ability to report TPM capabilities and to accept updates to PCRs. When enabled, all features of the TPM are available..
- Activated / Deactivated – Deactivation is similar to disabled, but operational state changes are possible. (i.e. change owner or activation with physical presence). When activated all features of the TPM are available.
- Owned / Un-owned – A platform is owned when an EK exists and the true owner knows owner authorization data. The owner of a platform may perform all operations including operational state changes.

These modes are mutually exclusive but may have overlapping influences. In cases where TPM commands are available by one mode and unavailable by another mode, precedence is given to the making the command unavailable.

A number of TPM commands control policies surrounding administration of operational state. These commands are guarded by state variables that limit access to commands that manipulate operational state. Some commands, for example, require an operator to be physically present at the platform - using physical controls to effect a change to platform's activation, enablement and ownership state.

TCG recommends a default operational mode be configured by the platform manufacturer. The recommendation places the system in a safe but flexible state for configuring an owner. The table in Figure 4:h shows the recommended settings.

| Operating State | Default Value | Change via Remote Operation |
|---|---|---|
| Enabled (by Owner)[7] | False | Yes |
| Enabled (by anyone) [8] | False | No |
| Ownership Enabled[9] | True | No |
| Owned | False | No |
| Activated – Persistent | False | No |
| Activated – Temporal | False | No |

Figure 4:h – Suggested default operational mode set by platform manufacturer.

The table also shows the operating states and input control states that require physical presence in order to effect operational mode changes. Essentially, all state changes require physical presence except state changes performed by the platform owner.

A platform's TPM can be temporarily deactivated. It can be deactivated while a sequence of keys on the keyboard are pressed and reactivated when the system is reset. Temporal activation does not change the

[7] Disabled and only the platform owner may enable the platform – contingent on there being an owner.
[8] Disabled and anyone may enable the platform.
[9] This controls TPM commands that allow a platform owner to be created.



IPR2025-01572
Apple EX1048 Page 28

persistent activation value. It is used by operators desiring to deactivate the TPM for short-lived interactions.

#### 4.5.1.1 Platform Deployment Scenarios

The following platform deployment scenarios help illustrate possible TPM modes and the circumstances in which mode changes might be applied. Assume all platforms are shipped with the recommended default configuration.

##### 4.5.1.1.1 An IT Owned and Managed Platform

The Asset Management team within a corporate IT organization receives the platform and begins tracking the asset. An Endorsement Key is generated by the vendor. Asset Management takes delivery of the platform and executes the TPM_TakeOwnership command and in so doing chooses the owner authorization secret. The authorization data and EK public key (e.g. Endorsement Credential) are included in an asset control record created for tracking the platform asset. No further ownership changes are anticipated, hence the ability to change platform ownership is disabled. The ownership-enabled flag is switched off. Asset Management knows the EK was not spoofed because they were able to take ownership initially.

Upon leaving Asset Management, the operational state has changed:

- Enabled (by owner) = False
- Enabled (by anyone) = False
- Ownership Enabled = **False** : (Changed)
- Owned = **True** : (Changed)
- Activated – Persistent = False
- Activated – Temporary = False

The platform is delivered to the site where it will be connected to an enterprise network. Once connected, the platform is enabled (remotely by Asset Management). A departmental technician activates the TPM in addition to configuring the software and management features. The TPM operational state changes as follows:

- Enabled (by owner) = **True** : (Changed)
- Enabled (by anyone) = False
- Ownership Enabled = False
- Owned = True
- Activated – Persistent = **True** : (Changed)
- Activated – Temporary = False

The platform is delivered to the intended user ready to operate. The TPM is fully functional; takes measurements and reports platform configuration. IT periodically monitors the platform for compliant configurations.

##### 4.5.1.1.2 A Consumer Owned Platform

A dealer sells a platform to a customer. The dealer helps the customer take ownership (in the case of an unsophisticated customer) and is careful not to learn or remember the owner pass phrase. The customer chooses not to enable the TPM. The operational state remains the same except ownership has been transferred to the customer. The ending operational state is as follows:

- Enabled (by owner) = False
- Enabled (by anyone) = False
- Ownership Enabled = **False** : (Changed)
- Owned = **True** : (Changed)

- Activated – Persistent = False
- Deactivated – Temporary = False.

#### *4.5.1.1.3 A Consumer Owned Platform with Outsourced Management*

A dealer sells the platform bundled with a service contract. Service is outsourced to a company specializing in servicing this particular platform. The service provider takes ownership and configures the platform to be remotely managed. The activation state is left to the discretion of the customer. Hence the following settings:

- Enabled (by owner) = **True** : (Changed)
- Enabled (by anyone) = False
- Ownership Enabled = False
- Owned = **True** : (Changed)
- Activated – Persistent = False
- Deactivated – Temporary = False.

The consumer uses TPM functionality only when interacting with the outsourced management company. He resets the platform and activates the TPM when manageability functions are performed. If the consumer wishes to use the Internet anonymously he may temporarily deactivate the TPM by pressing control keys on the keyboard. The following settings are in force:

- Enabled (by owner) = True
- Enabled (by anyone) = False
- Ownership Enabled = False
- Owned = True
- Activated – Persistent = **True** : (Changed).
- Deactivated – Temporary = **True** : (Changed).

Management activity is placed on hold while the customer accesses the Internet then resumes when the customer restarts his machine and activation state reverts to the persistent value (activated).

The customer may select a different management service provider once the initial contract expires. Hence, the Ownership Enabled state must be reset to True. This action could be performed by the customer or the management service.

The scenarios presented here are only a few of the many possible configuration options. Different platform deployment models may suggest which different operational settings are best. The important point is that distribution and deployment mechanisms play a role in ensuring platform safety and user privacy requirements are met.

### 4.5.2 Platform Operation

#### *4.5.2.1 System Startup and Initialization*

When the platform begins operation, it is required to start at the RTM. The RTM is aware of itself and the other trusted building blocks (i.e. RTS and RTR). The TPM starting state is equivalent to system initialization state. When power-on startup occurs, the RTM is required to signal the TPM[10] instructing it to start its initialization process. The TPM initialization process includes a TPM self-test. The self-test determines if the TPM is functioning properly.

Different power management modes may impact the TPM. The RTM is responsible for selecting and controlling the most appropriate TPM initialization processes. The TPM will respond accordingly. The following power management modes are anticipated for TPM devices:

[10] Depending on implementation decisions, the RTM may not actually reside within the TPM.

- Initial power-up – The TPM must set all PCR registers to zero and reset control flags
- Sleep – The TPM must save volatile state to persistent storage
- Resume – The TPM must recover saved state
- Hibernate – The TPM doesn't have a corresponding power state. Volatile state must be saved.

Operations such as these can be very platform specific. It is the responsibility of the Platform Specific Specification(s) to fully document valid platform states and TPM responses.

As part of system initialization, measurements of platform components and configurations will be taken. Taking measurements will not detect unsafe configurations nor will it take action to prevent continuation of the initialization process. This responsibility rests with a suitable reference monitor such as an operating system.

#### *4.5.2.2 Application Loading*

The operating system program loader is the next logical soft component to measure a program prior to loading it. Since the operating system helps enforce system integrity, it is reasonable for the program loader to both measure and enforce policies describing unacceptable software configuration state.

Applications may contain policies describing trusted platform configurations and refuse or limit interaction with the platform as a consequence. Such bilateral measurement semantics may be extended to other platforms, thereby enabling distributed application measurement and enforcement. Application peers may entertain cross-checking prior to engaging in collaborative exchange as a safety precaution.

### 4.5.3 Interfacing with TPM and Software Services

#### *4.5.3.1 Soft Interfaces and Services*

There are three interfaces envisaged for TCG software. They correspond to services layering common to most general purpose computing platforms. Figure 4:i shows the TPM and low-level driver at the kernel mode level. The majority of the TCG TPM specification documents this interface.



IPR2025-01572
Apple EX1048 Page 31

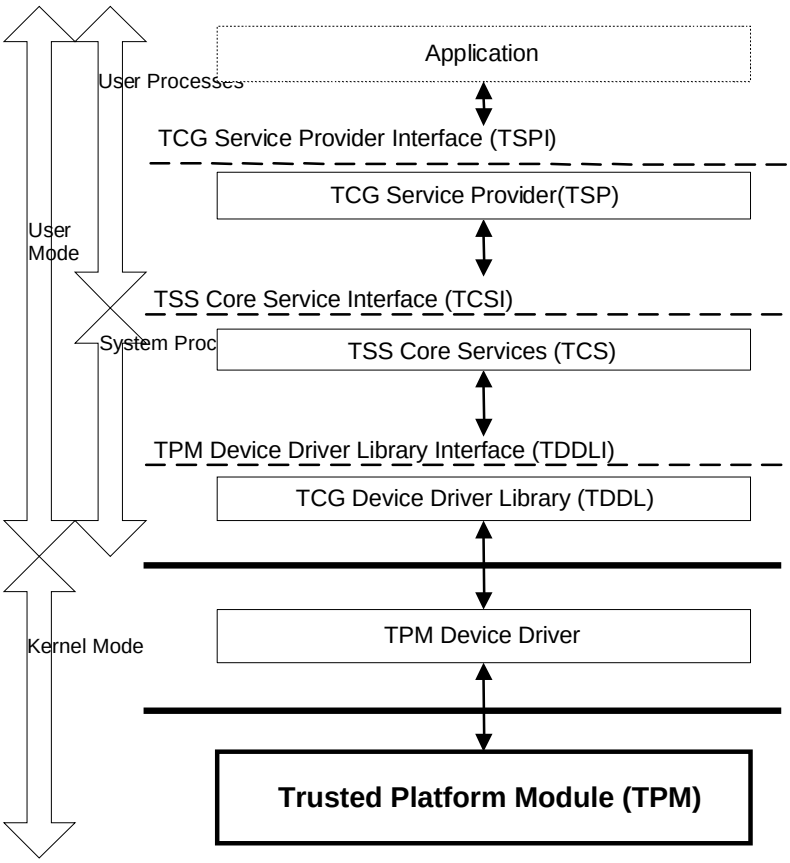



Figure 4:i – TCG Software Layering.

#### *4.5.3.1.1 TDDL Interface*

Progressing upward from the TPM device and device driver, the diagram (Figure 4:i) shows the TDDLI which is a user mode interface. Such an interface has several advantages over a kernel mode driver interface:

- It ensures different implementations of the TCG software stack properly communicate with any TPM.
- It provides an OS-independent interface for TPM applications.
- It allows the TPM vendor to provide a software TPM simulator as a user mode component.

The TDDL provides the transition between user mode and kernel mode. It does not manage threaded interactions with the TPM, nor does it perform TPM command serialization. These are applied higher in the stack. Since the TPM is not multithreaded, there would be a single-instance of the TDDL, per platform, and it enforces single threaded access to the TPM.

#### *4.5.3.1.2 TCS Interface*

The TCG Core Services (TCS) provides an interface to a common set of platform services. Though there may be multiple TCG Service Providers on a single platform, the TCS ensures they all exhibit common behavior. The TCS provides five core services:

- Context Management – Implements threaded access to the TPM
- Credential & Key Management – Stores credentials and keys associated with the platform
- Measurement Event Management – Manages event log entries and access to associated PCR registers

- Parameter Block Generation – responsible for serializing, synchronizing and processing TPM commands

The TCS operates as a system process in user mode. It is trusted to manage authorization information supplied to the TPM.

#### *4.5.3.1.3 TSP Interface*

The TCG Service Provider (TSP) exposes a C interface to the TPM, based on an object oriented underlying architecture. It resides within the same process address space as the application. Authorization takes place at this layer, either using a user interface coded to this layer or via a callback mechanism at the TCS layer (if the caller is remote). In order to provide a consistent authorization interface to the end user, local applications do not provide authentication services, but rather rely on that inherent in the platform.

The TSP provides two services, *context management* and *cryptography*. The Context Manager provides dynamic handles that allow for efficient usage of application and TSP resources. Each handle provides context for a set of interrelated TCG operations. Different threads within the application may share the same context or may acquire a separate context per thread.

To make full use of TPM protected functions, supporting cryptographic functions must be provided. The TSP does not provide this support except as is necessary to perform operations required by the specification. In particular, bulk data encryption is not exposed by the interface. Example functions include message digesting and byte-stream generation.

### *4.5.3.2 Application Interaction Scenarios*

It is anticipated that TCG services will be used by new and existing applications and application infrastructures. This section highlights ways in which common application architectures might integrate TCG functionality. Figure 4:j depicts three scenarios. The first scenario, shows the straight forward interaction between the TPM and an application. The second shows interaction through an existing infrastructure that provides cryptographic and security services. The third is an example illustrating remote access through an RPC mechanism.

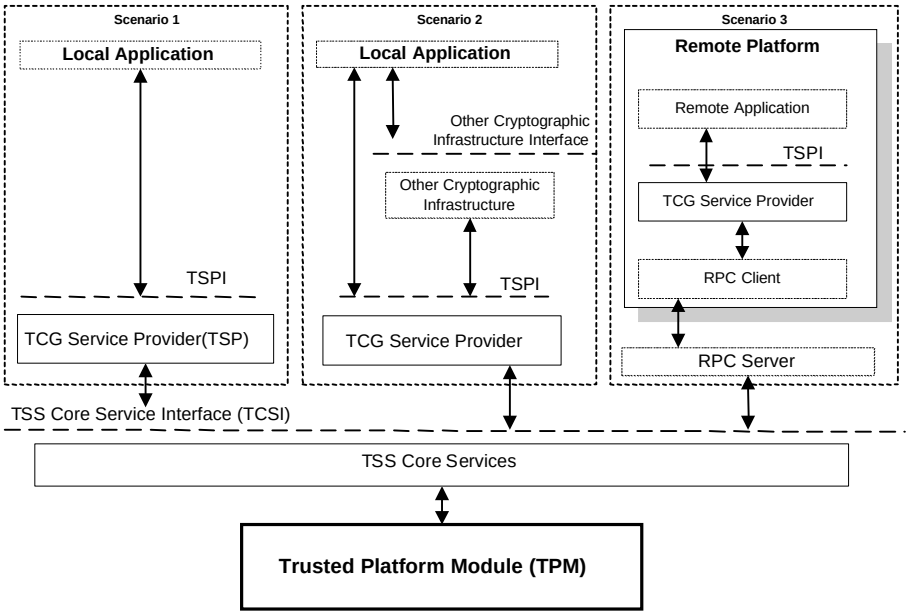


Figure 4:j – Application Interaction Scénarios.

#### *4.5.3.2.1 Scenario 1 – Comparing Measurement Events*

This scenario traces the calling sequence of an application verifying platform configuration contained in TPM managed PCR registers with expected values contained in Validation Credentials. The following steps are involved:

1) Initialize application objects and prepare to read PCR registers.
2) Read PCR5 value.
3) Compare PCR value(s) with validation values.

Application:

```
{
Tspi_Contest_Create( &hContext);
Tspi_Context_Connect(hContext, NULL);
Tspi_Context_GetTpmObject(hContext,&hTPM);
Tspi_TPM_PcrRead(hTPM, 5, &ulPcrValueLength, &rgbPCRValue);
Compare(correctValueOfPCR5, rgbPCRValue);
}
```

TSP:

When Tspi_TPM_Quote() is called, the TCS requires the key used to sign PCR values to be loaded into the TPM. This would be done with the Tspi_Context_LoadKeyByUUID for example.

It also requires a PcrComposite index to have been generated which identifies the PCR values that are to be quoted. This is done with the Tspi_PcrComposite_SelectPcrIndex command.

When executed, the command will call the Tcsip_Quote command.

The Tcsip_Quote() function signs and retrieves PCR values.

```
{ …
hKey = Tcsip_LoadKeyByBlob(ikey); //Load the key identified by its hash
Tcsip_Quote(hKey,…);              //Retrieve signed PCR
… }
```

The Tcs command in turn will call the device driver through the Tddl_ command.

TCS:

```
{ …
loadKeyMsg = PBG_LoadKey(hKey); //Marshall TPM_LoadKey command
quoteMsg = PBG_Quote();         //Marshall TPM_Quote command
Tddli_Open();                   //Open TPM communications channel
Tddli_TransmitData(loadKeyMsg); //Send/Recv response
Tddli_TransmitData(quoteMsg);   //Send/Recv response
Tddli_Close();                  //Optionally close channel w/ TPM
… }
```

TPM:

Upon receiving the messages to load a key (loadKeyMsg) and retrieve PCR values (quoteMsg), the TPM parses the command blocks sequentially and performs the appropriate operation.



IPR2025-01572
Apple EX1048 Page 34

#### 4.5.3.2.2 Scenario 2 – TPM as a Fixed Token Storage Device

In Scenario 2, existing interfaces provide fixed-token / smartcard storage capabilities (e.g. PKCS11[11]) that may be leveraged to access the TPM device for storage / retrieve of symmetric / asymmetric keys.

#### 4.5.3.2.3 Scenario 3 – Reading a PCR from a Remote Platform

In Scenario 3, the application interacts with a TPM that is remotely connected. The TCS implementation is built using an remote procedure call (RPC) or other messaging service. The current specification only specifies how to use a COM interface to talk to the remote system, but it is anticipated that other RPC mechanisms will be available. The remote application in this case will talk directly to the TCS. After the TCS Parameter Block Generator service marshals TPM messages (loadKeyMsg, quoteMsg), they are prepared for RPC transmission - rather than TDDL transmission.

Remote TCS:

```
{ …
loadKeyMsg = PBG_LoadKey(hKey); //Marshall TPM_LoadKey command
quoteMsg = PBG_Quote();         //Marshall TPM_Quote command
RPC_Open();                     //Open TPM communications channel
RPC_Send(loadKeyMsg);           //Send/Recv response
RPC_Send(quoteMsg);             //Send/Recv response
RPC_Recv(loadKeyMsg);           //Send/Recv response
RPC_Recv(quoteMsg);             //Send/Recv response
RPC_Close();                    //Optionally close channel w/ TPM
…}
```

Upon receipt, the RPC engine delivers TPM messages to the TCS to be processed locally.

Local TCS:

```
{ …
RPC_Recv(loadKeyMsg);           //Recv command
RPC_Recv(quoteMsg);             //Recv command
Tddli_Open();                   //Open TPM communications channel
Tddli_TransmitData(loadKeyMsg);  //Send/Recv response
Tddli_TransmitData(quoteMsg);   //Send/Recv response
Tddli_Close();                  //Optionally close channel w/ TPM
RPC_Send(loadKeyMsg);           //Send reply
RPC_Send(quoteMsg);             //Send reply
…}
```

These scenarios demonstrate that TPM functionality can be reached from local applications, local applications with mobile components and remote applications. TCG software layering allows the TPM to present a consistent set of services and interfaces while accommodating the rich variety of application environments.

### 4.5.3.3 TPM Command Validation

All commands to the TPM that affect security, privacy or reveal platform secrets must be authorized. Authorization means the caller must supply a secret as part of command invocation.

Several commands do not require authorization. They fall into two categories:

- Informational commands (i.e., those which contain no security or privacy information),
- Privacy relevant meta commands (i.e. those needed to configure command validation).

An example of an Informational command is the TPM_GetCapability function. This function retrieves the TPM manufacturing information like model name and part number. It does not include unique identifiers such as serial number, Key ID or Platform ID.

[11] PKCS#11 Cryptographic Token Interface Standard; http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/

#### *4.5.3.3.1 How Command Validation Works*

Any *Entity* (process, thread or embedded controller) may submit TPM commands. Entities and the TPM form a secure communications channel through which TPM commands are submitted and results are returned. The channel follows request-response semantics for session-oriented message exchange.

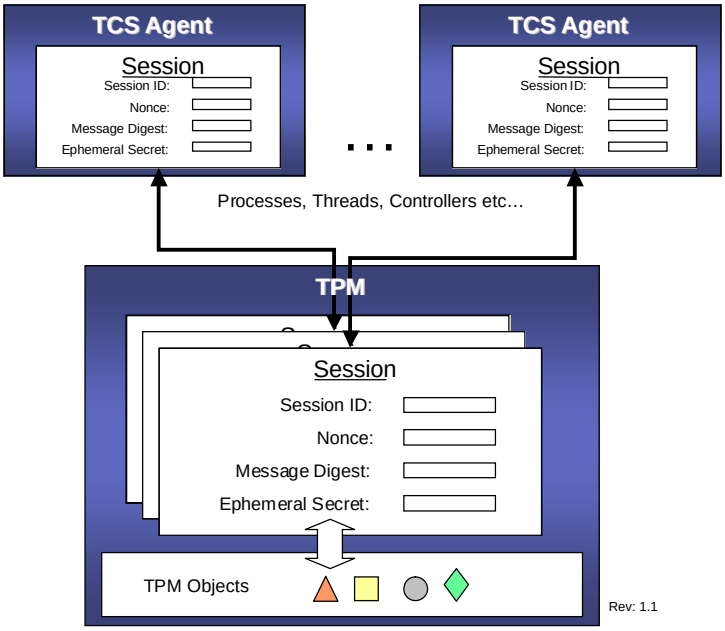



Figure 4:k - Command Validation Sessions and Endpoints

Figure 4:k shows the TPM managing several Command Validation sessions with external entities. A session consists of a unique session identifier, common to session endpoints; nonces (different for each endpoint); a digest of messages exchanged between endpoints (updated after each exchange) and lastly ephemeral secrets used to tie messages exclusively to a named object and optionally to encrypt message traffic.

The purpose for session establishment is to ensure access to TPM objects is authorized. To participate in authorized exchange, entities must supply a pass-phrase which both authenticates and authorizes them to use it. The *authorization secret*, is a 160-bit value (ideally random and non-guessable). There is no "required" method for generating the secret however. The size was chosen to match that of a SHA-1 operation result. The assumption being secrets, salts and any other values will be hashed to produce a fixed sized result – called *authorization data*.

Authorization data may be associated with TPM objects (such as keys, BLOBS and discrete components), the TPM itself or command interfaces.

The command validation protocols use HMAC, as defined by RFC 2104[12], to create an authorized *session* between caller and the TPM. A message in an authorized session consists of a frame containing three parts.

- Message Container – identifies message type, size and formatting
- TPM Command – command name input/output parameters and return code
- Session State – session ID, control flags and digest value of session messages

Both TPM and caller validate session messages prior to moving to the next step in the request-response protocol. Session replay is avoided by including rolling-nonces in the session state.

The number of distinct sessions the TPM supports and whether or not there may be simultaneous sessions is left as an implementation decision. However, TCG mandates that request-response pairs be

[12] http://www.ietf.org/rfc/rfc2104.txt?number=2104

atomic exchanges and at least 3 sessions are needed for some commands. The TPM will not accept a new request until any pending reply is processed.

It is important to emphasize that the security properties of command validation protocols have been specifically designed **not** to rely on security properties of the data transport. None of the TSS modules, RPC communications or other components should affect the trusted properties of the TPM. All modules, components and interfaces outside the TPM are considered *un-trusted* in relation to the TPM, but tampering is detectable by the TPM.

#### *4.5.3.3.2 Protocols that Support Command Validation*

TCG defines five protocols that implement command validation primitives. ADIP, ADCP and AACP are used to create and manage authorization information, which is contained in objects under the control of the TPM. Additionally OIAP and OSAP are used to establish authorized session contexts leveraged by the other protocols.

ADIP, ADCP and AACP semantics are tightly integrated with TPM management commands. A generalized interface is not exposed by the TPM.

OIAP and OSAP are accessed like other TPM commands, through the TPM command interpreter. There is a TPM command interface and serialization method defined for each. See Section 0 for more detail on Command Serialization. TPM_OIAP() and TPM_OSAP() commands are used to initialize session objects. A handle to the session object is included with other TPM commands that require validation.

##### *4.5.3.3.2.1 Object-Independent Authorization Protocol (OIAP)*

The OIAP protocol establishes an authorized clear-text session between the TPM and an external entity. The TCG Core Services (TCS) library provides useful features for managing OIAP sessions.

A sequence of OIAP message exchanges and corresponding TCS/TPM operations is depicted in Figure 4:l. Three subjects are depicted (TCS, OIAP Session and TPM). The TCS and OIAP objects may reside on a local or remote platform relative to the TPM object. TCG attention focuses on protection mechanisms suitable for the TPM endpoint, but recognizes additional mechanisms are required to secure the TCS endpoint. The salient OIAP exchanges occur between OIAP Session and TPM. TCS exchanges provide context describing how OIAP Sessions might be employed.

Message flows (1-5) depict OIAP session establishment. An established session can be used multiple times to wrap TPM commands – depicted in flows (6-15). The TCS agent determines when there are no more commands to issue and sets a flag on the last command execution message triggering session cleanup – see flows (16-17). The OIAP Session object is temporal in that its context goes out of scope when the last command has been processed.

OIAP ensures message integrity by calculating a message authentication code (MAC) for all messages exchanged. OIAP uses HMAC, which takes a secret and a hash of the message data as input and produces a 20 byte digest. OIAP defines two nonce words, an *even* nonce and an *odd* nonce; corresponding to session endpoints. This approach ensures neither side is dependent on the other for entropy. Flows (4) and (9), in Figure 4:l, are used to exchange respective nonce values. Flow (4) also contains a session identifier.

The other element of a MAC is a hash of the message data. Together, the hash of secret and message data forms the session MAC. A session is comprised of two logical structures, 1) TPM command – including parameters and return code. A SHA-1 hash result of the TPM Command fields is what is actually included in the computation of the MAC. 2) Session setup parameters – including session ID, nonce words and control flags. Session setup parameters are included in HMAC computation.

OIAP protocol and protocol data units are explained in more detail in the TCG 1.2 Design Philosophy document.



IPR2025-01572
Apple EX1048 Page 37

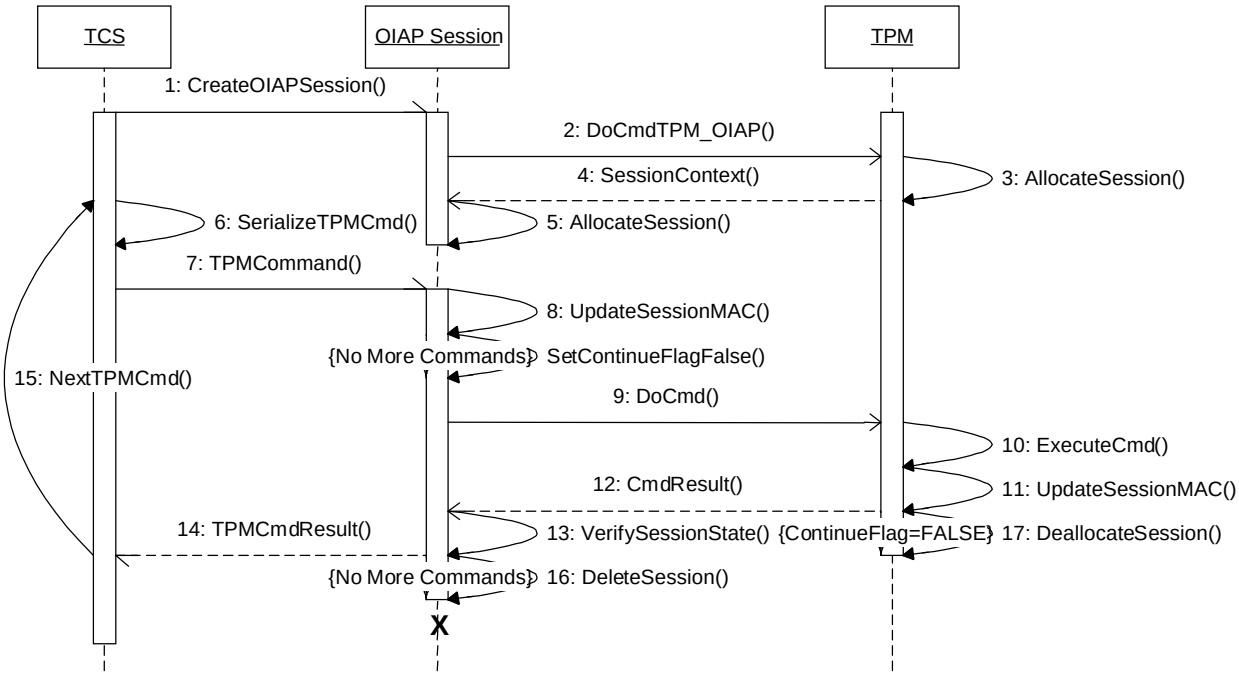



Figure 4:l - OIAP Sequence

There is no expiration or timeout associated with sessions. Hence, the time taken to determine the next TPM Command (flow 15 in Figure 4:l) may be infinite. A denial of service consequence may result when TPM session structures are exhausted.

Sharing the session endpoint (i.e. TCS Object in Figure 4:l) may introduce security concerns, as the TCS object must also control access to the session object and protect clear-text shared-secrets. TCG recognizes a vulnerability exists, but remains confident that security designers can find suitable mechanisms to counter the vulnerability. TCG also recognizes that system security is achieved only when all system components behave appropriately and stresses the TCG goal to provide building-blocks rather than complete systems.

#### *4.5.3.3.2.2 Object-Specific Authorization Protocol (OSAP)*

OSAP is very similar to OIAP in concept and design. It constrains OIAP semantics in that the authorized session is bound to a TPM object and it computes an ephemeral secret. Protocol flow differs from OIAP in two ways other than the differing command ordinals for OIAP and OSAP.

- In flow (2) Figure 4:l, the target TPM object is identified and a third nonce (in addition to OIAP nonce words) is supplied.
- In flow (4), a fourth nonce is supplied.

The additional nonce words are used to generate a session ephemeral secret . This secret is used to calculate the MAC and may be used to encrypt data.

OSAP is particularly useful for updating sensitive information associated with TPM managed objects.

#### *4.5.3.3.2.3 Authorization Data Insertion Protocol (ADIP)*

The ADIP protocol is used when a caller desires to instantiate new TPM managed objects. ADIP allows the caller supplied pass phrase to be associated with the new object. The ADIP protocol leverages OSAP to build an authorized session with the parent of the new object (A.K.A. child).

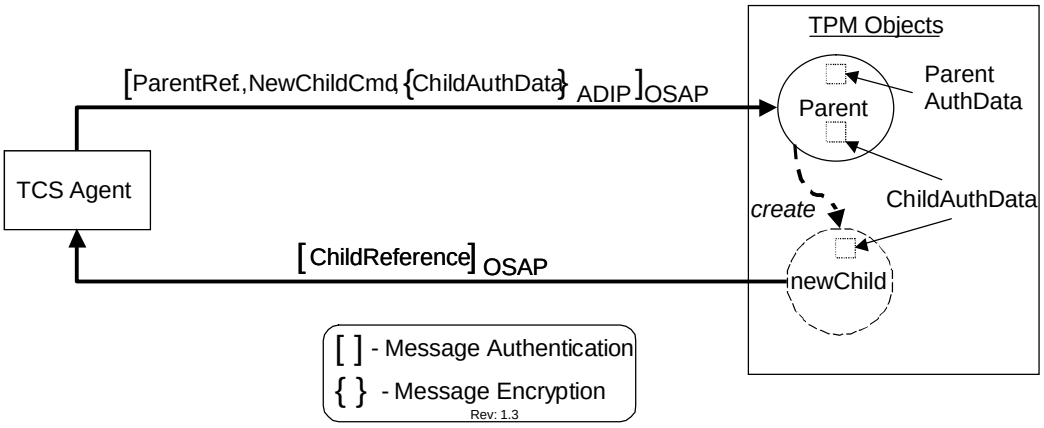



Figure 4:m - Object Creation Using ADIP

The child object's authorization data is encrypted using the OSAP shared secret by XOR. The TPM decrypts authorization data by XOR of cipher text with the shared secret.

TPM_CreateWrapKey() uses the ADIP protocol.

#### *4.5.3.3.2.4 Authorization Data Change Protocol (ADCP)*

The ADCP protocol is used to update authorization data for TPM managed objects. ADCP leverages ADIP for privacy and integrity and adds an additional OIAP (or OSAP) session for authorized access to the child.

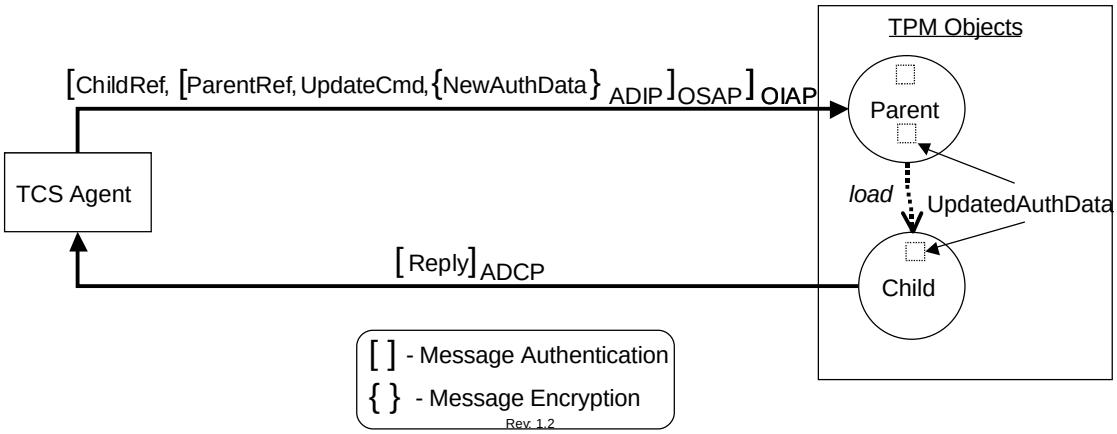



Figure 4:n - Updating Child Authorization Data using ADCP

Most objects managed by the TPM fit the parent-child relationship model. There are some exceptions however. The SRK logically is the root of the storage sub-system. The TPM can accommodate ADCP semantics by using the EK authorization data to establish the ADIP session for the SRK.

The TPM_ChangeAuth() command uses ADCP protocol.

#### *4.5.3.3.2.5 Asymmetric Authorization Change Protocol (AACP)*

The AACP protocol allows a child's authorization data to be changed without the parent learning the child's authorization data. AACP prevents a threat in ADCP that exists when the parent authorization data is shared between multiple entities and some of those entities are not authorized to access the shared-parent's child objects. This case is likely to occur for the immediate children of the SRK.



IPR2025-01572
Apple EX1048 Page 39

The AACP protocol is used by TPM_ChangeAuthAsymStart() and TPM_ChangeAuthAsymFinish() commands.

## 4.6 TCG Programmatic Interfaces

### 4.6.1 Naming Conventions

TCG specifications follow a stylized convention when referring to functionality of various components. Terms such as *command*, *function*, *operation* and *interface* are frequently encountered.

- Command – discrete functionality of the TPM exposed externally and recognizable by TPMs command processor. TPM commands are enumerated and have an ordinal value associated.
- Function – discrete functionality of non-TPM modules having programmatic interfaces.
- Operation – refers to a sequence of steps or protocol flow that may be implemented using one or more commands or functions.
- Interface – The set of command or function entry points, including parameters and return codes, to a particular module. When used in singular context, Interface may refer to a single entry point.

TCG operations are classified according to security relevance. They are:

- Protected Operations – Operations affecting the security properties of TCG platforms. These include all TPM commands. TPM command interface names begin with the “TPM_” prefix.
- Unprotected Operations – Operations that support TPM protected functionality, but are not TPM commands. These are normally implemented outside the TPM. Function interface names begin with Tspi_, Tcsi_,or Tddli_ prefix.
- Connection Operations – Operations involving platform to TPM binding. Commands implementing connection operations are typically defined in the *Platform-Specific* specifications. These command interface names begin with the “TSC_” prefix. For example, TSC_PhysicalPresence() includes functionality defined only in the platform-specific specification. TSC stands for “TCG Software Connection”.

TCG functions that extend, layer or encapsulate TPM functionality are prefixed with the module name. For example, TCG Core Services interfaces have the “Tcsi_” prefix; TCG Service Provider interfaces have the “TSPI_” prefix; and TCG Device Driver interfaces have the “Tddli_” prefix.

### 4.6.2 Command Ordinals & Serialization

The TPM command interface employs a serialized request-response model of interaction similar in concept to Remote Procedure Call (RPC). A command interpreter / parameter-block generator pair is used to encode/decode command requests and responses. The TCS provides such services for non-TPM entities. The TPM I/O Component (See section 4.3) provides these services for the TPM.

Command messages have a message header that identifies message type (a.k.a. *tag*) and message size (in bytes). Message types are either *request* or *response* and specializations for two types of authorized request/response messages exist. There are 6 message types in all.

Request/Response Message:

| Message Type Tag | Message Size (*includes Tag & Size*) |
|---|---|
| *2 Bytes* | *4 Bytes* |

TPM Command syntax is expressed in serialized notation. Each command has an ordinal value which uniquely identifies a TPM entry point and interface semantics. The command ordinal is followed by a parameter block and return code (for reply messages). The ordinal also acts as a magic number for content formatting. Changes in interface semantics would necessarily result in an ordinal value change.

Command Call:



IPR2025-01572
Apple EX1048 Page 40

| Command Ordinal | Parameter Block (Param1… Param-n) |
|---|---|
| *4 Bytes* | *(determined by ordinal semantics)* |

Command Reply:

| Return Code | Parameter Block (Param1… Param-n) |
|---|---|
| *4 Bytes* | *(determined by ordinal semantics)* |

Command invocation is atomic between request and response. No other callers may submit commands while a command reply is pending within the TPM.

If a message is authorized (ala OIAP, OSAP) the message also contains a trailer. The trailer includes a MAC digest, session identifier and nonce words.

Session Trailer - OSAP Call:

| Session Handle | Session Nonce (caller) | Flags | Session MAC (caller) |
|---|---|---|---|
| *4 Bytes* | *20 Bytes* | *1Byte* | *20 Bytes* |

Session Trailer – OSAP Reply:

| Session Nonce (TPM) | Flags | Session MAC (TPM) |
|---|---|---|
| *20 Bytes* | *1Byte* | *20 Bytes* |

Binary values are serialized in *network-byte-order*. Character data is in ASCII.

### 4.6.3 Summary of TCG Commands and Interfaces

This section contains summaries of TCG interfaces consisting of TPM Commands, TCG Device Driver Library (TDDL), TCG Core Services Interface (TCS) and TCG Service Provider Interface (TSP).

#### *4.6.3.1 TPM Commands*

| Component | Area | Command Name | Description |
|---|---|---|---|
| TPM / RTS | Protected Storage Commands | TPM_Seal<br>TPM_Unseal<br>TPM_UnBind<br>TPM_CreateWrapKey | These commands use public-key cryptography to prepare arbitrary data and keys for private key operations at TPM endpoints, and to perform those private key operations. TPM endpoints may be explicitly refined via platform configuration register values. |
| | Key Management Commands | TPM_LoadKey<br>TPM_EvictKey<br>TPM_GetPubKey<br>TPM_CertifyKey<br>TPM_SaveKeyContext<br>TPM_LoadKeyContext | These commands control which keys are available for use by the TPM and prepare keys for safe storage outside the TPM package. |
| | Migration Commands | TPM_CreateMigrationBlob<br>TPM_ConvertMigrationBlob<br>TPM_AuthorizeMigrationKey | These commands are used to transfer migratable objects from one TPM to another. |
| | Maintenance Commands (optional) | TPM_CreateMaintenanceArchive<br>TPM_LoadMaintenanceArchive<br>TPM_KillMaintenanceFeature<br>TPM_LoadManuMaintPub<br>TPM_ReadManuMaintPub | These commands are used to transfer non-migratable objects from one TPM to another. Transfer requires cooperation of both the TPM owner and an external entity, probably the platform OEM or their agent. |



IPR2025-01572
Apple EX1048 Page 41

| Component | Area | Command Name | Description |
|---|---|---|---|
| TPM / RTM | Measurement Collection Commands | TPM_Extend<br>TPM_DirWriteAuth<br>TPM_SHA1Start<br>TPM_SHA1Update<br>TPM_SHA1Complete<br>TPM_SHA1CompleteExtend | These commands facilitate update of Platform Credential Register (PCR) and Data Integrity Register (DIR) values and for computing hash and extend values |
| TPM / RTR | Measurement Reporting Commands | TPM_PcrRead<br>TPM_Quote<br>TPM_DirRead<br>TPM_DirReadSigned | These commands facilitate reporting of Platform Credential Register (PCR) and Data Integrity Register (DIR) values. These commands may use AIK keys. |
| | TPM Endorsement Key Commands | TPM_CreateEndorsementKeyPair<br>TPM_ReadPubek<br>TPM_DisablePubekRead<br>TPM_OwnerReadPubek | These commands manipulate the platform Endorsement Key (EK) and manage access control policy. |
| | AIK Commands | TPM_MakeIdentity<br>TPM_ActivateIdentity | These commands manage the creation, activation and recovery of Attestation Identity Keys (AIK). |
| TPM Support Services | Authentication Protocols and Authorization Commands | TPM_OIAP<br>TPM_OSAP<br>TPM_ChangeAuth<br>TPM_ChangeAuthOwner<br>TPM_ChangeAuthAsymStart<br>TPM_ChangAuthAsymFinish<br>TPM_SaveAuthContext<br>TPM_LoadAuthContext | These commands establish authorized sessions for exchanging commands with the TPM. They also manage access controlled objects contained within the TPM. |
| TPM Misc. Services | Cryptographic Commands | TPM_Sign<br>TPM_GetRandom<br>TPM_StirRandom | These commands provide general purpose cryptographic services. |
| | Auditing Commands | TPM_GetAuditEvent<br>TPM_GetAuditEventSigned<br>TPM_SetOrdinalAuditStatus<br>TPM_GetOrdinalAuditStatus | These commands are used to collect audit trail data and control auditing features. |
| | Capability Reporting Commands | TPM_GetCapability<br>TPM_GetCapabilitySigned<br>TPM_GetCapabilityOwner | These commands provide information about the TPM part and implemented functionality. |
| TPM Manageme nt | TPM Ownership Commands | TPM_TakeOwnership<br>TPM_SetOwnerInstall<br>TPM_OwnerSetDisable<br>TPM_FieldUpgrade<br>TPM_SetRedirection | These commands are used to initialize the TPM for deployment and for field maintenance. |
| | Operational Flags Commands | TPM_OwnerClear<br>TPM_DisableOwnerClear<br>TPM_ForceClear<br>TPM_DisableForceClear<br>TPM_PhysicalDisable<br>TPM_PhysicalEnable<br>TPM_PhysicalSetDeactivated<br>TPM_SetTempDeactivated<br>TSC_PhysicalPresence | These commands configure the operational modes of the TPM. |
| | Self-Test Commands | TPM_SelfTestFull<br>TPM_CertifySelfTest<br>TPM_ContinueSelfTest<br>TPM_GetTestResult | These commands are used to detect and diagnose problems with TPM operation. |

| Component | Area | Command Name | Description |
|---|---|---|---|
| | Startup Commands | TPM_Reset<br>TPM_Init<br>TPM_SaveState<br>TPM_Startup | These commands are used to reset and restart the TPM. |

Figure 4:o - TPM Commands Summary

### 4.6.3.2 TCG Device Driver Library Interfaces

| Component | Area | Command Name | Description |
|---|---|---|---|
| TSS / TDDL | Device Driver Library Interfaces | Tddli_Open<br>Tddli_Close<br>Tddli_Cancel<br>Tddli_GetCapability<br>Tddli_SetCapability<br>Tddli_GetStatus<br>Tddli_TransmitData | These functions manage low-level interaction with the TPM device driver. They are used to prepare the channel for exchange of TPM command blocks, move data through the channel and report on the condition of the channel. |

Figure 4:p - TDDL Functions Summary

### 4.6.3.3 TCG Core Services Interfaces

| Component | Area | Command Name | Description |
|---|---|---|---|
| | TPM Ownership, Authorization and Identity | Tcsip_SetOwnerInstall<br>Tcsip_TakeOwnership<br>Tcsip_OIAP<br>Tcsip_OSAP<br>Tcsip_ChangeAuth<br>Tcsip_ChangeAuthOwner<br>Tcsip_ChangeAuthAsymStart<br>Tcsip_ChangeAuthAsymFinishTcsip_TerminateHandle | These functions expose TPM functionality for accessing and controlling objects managed by the TPM. |
| | PBG Functions<br>- TPM Mandatory | Tcsip_Extend<br>Tcsip_PcrRead<br>Tcsip_Quote<br>Tcsip_DirWriteAuth<br>Tcsip_DirRead<br>Tcsip_Seal<br>Tcsip_Unseal<br>Tcsip_UnBind<br>Tcsip_CreateMigrationBlob<br>Tcsip_ConvertMigrationBlob<br>Tcsip_AuthorizeMigrationKey | These functions expose TPM functionality for moving objects between TPMs, interfacing with PCR and DIR registers. |
| | PBG Functions TPM Cryptographic Capabilities | Tcsip_CertifyKey<br>Tcsip_Sign<br>Tcsip_GetRandom<br>Tcsip_StirRandom<br>Tcsi_GetCapability<br>Tcsip_GetCapabilityOwner | These functions expose TPM cryptographic functionality. |



IPR2025-01572
Apple EX1048 Page 43

| Component | Area | Command Name | Description |
|---|---|---|---|
| | PBG Functions TPM Endorsement Credentials | Tcsip_CreateEndorsementKeyPair<br>Tcsip_ReadPubek<br>Tcsip_DisablePubekRead<br>Tcsip_OwnerReadPubek | These functions expose TPM endorsement key functionality. |
| | PBG Functions - TPM Optional | Tcsip_CreateMaintenanceArchive<br>Tcsip_LoadMaintenanceArchive<br>Tcsip_KillMaintenanceArchive<br>Tcsip_LoadManufaturerlMaintenancePub<br>Tcsip_ReadManufacturerMaintenancePub | These functions expose TPM functionality for managing containers used to exchange TPM managed objects between TPMs. |
| | PBG Functions – TPM Self-test and Management | Tcsip_CertifySelfTest<br>Tcsip_GetTestResult<br>Tcsip_SelfTestFull<br>Tcsip_ContinueSelfTest<br>Tcsip_PhysicalPresTcsip_OwnerSetDisabl e Tcsip_OwnerClear<br>Tcsip_DisableOwnerClear<br>Tcsip_ForceClear<br>Tcsip_DisableForceClear<br>Tcsip_PhysicalDisable<br>Tcsip_PhysicalEnable<br>Tcsip_PhysicalSetDeactivated<br>Tcsip_SetTempDeactivated<br>Tcsip_FieldUpgrade<br>Tcsip_SetRedirection | These functions expose TPM management and administration functionality. |
| TCS Event Manager | Event Manager Functions | Tcsi_LogPcrEvent<br>Tcsi_GetPcrEvent<br>Tcsi_GetPcrEventsByPcr<br>Tcsi_GetPcrEventLog | These functions manage TCG Event Logs. |
| TCS Key & Credential Manager | Keys | Tcsi_RegisterKey<br>Tcsip_UnregisterKey<br>Tcsi_GetRegisteredKeyBlob<br>Tcsip_GetRegisteredKeyByPublicInfoTcsi_ EnumRegisteredKeys<br>Tcsi_GetRegisteredKey<br>Tcsip_LoadKeyByBlob | These functions help with the manipulation, storage and caching of objects containing keys. |
| | Credentials | Tcsip_GetPubKey<br>Tcsip_MakeIdentity | These functions help with the manipulation of signed documents that certify manufacturers and other vouching for TCG technology. |
| TCS Context Manager | Memory Management Functions | Tcsi_OpenContext<br>Tcsi_CloseContext<br>Tcsi_FreeMemory | These functions establish memory contexts useful to programs multiplexing access to the TPM. |
| TCS Audit Manager | Auditing Functions | | Currently not implemented. |

Figure 4:q - TCS Functions Summary

### 4.6.3.4 TCG Service Provider Interfaces

| Component | Area | Command Name | Description |
|---|---|---|---|



IPR2025-01572
Apple EX1048 Page 44

| Component | Area | Command Name | Description |
|---|---|---|---|
| TCG Service Provider (TSP) Interface | Common Methods | Tspi_SetAttribUint32<br>Tspi_GetAttribUint32<br>Tspi_SetAttribData<br>Tspi_GetAttribData<br>Tspi_ChangeAuth<br>Tspi_ChangeAuthAsym<br>Tspi_GetPolicyObject | These methods are common to all TSP classes. |
| | Context Class Methods | Tspi_Context_Create<br>Tspi_Context_Close<br>Tspi_Context_Connect<br>Tspi_Context_FreeMemory<br>Tspi_Context_GetDefaultPolicy<br>Tspi_Context_CreateObject<br>Tspi_Context_CloseObject<br>Tspi_Context_GetCapability<br>Tspi_Context_GetTPMObject<br>Tspi_Context_LoadKeyByBlob<br>Tspi_Context_LoadKeyByUUID<br>Tspi_Context_RegisterKey<br>Tspi_Context_UnregisterKey<br>Tspi_Context_DeleteKeyByUUID<br>Tspi_Context_GetKeyByUUID<br>Tspi_Context_GetKeyByPublicInfo<br>Tspi_Context_GetRegisteredKeysByUUID | These methods do container management for TPM managed objects. This includes caching and external object archival. |
| | Policy Class Methods | Tspi_Policy_SetSecret<br>Tspi_Policy_FlushSecret<br>Tspi_Policy_AssignToObject | These methods manage authentication and authorization policies for TPM managed objects. |
| | TPM Class Methods | Tspi_TPM_CreateEndorsementKey<br>Tspi_TPM_GetPubEndorsementKey<br>Tspi_TPM_TakeOwnership<br>Tspi_TPM_CollateIdentityRequest<br>Tspi_TPM_ActivateIdentity<br>Tspi_TPM_ClearOwner<br>Tspi_TPM_SetStatus<br>Tspi_TPM_GetStatus<br>Tspi_TPM_SelfTestFull<br>Tspi_TPM_CertifySelfTest<br>Tspi_TPM_GetTestResult<br>Tspi_TPM_GetCapability<br>Tspi_TPM_GetCapabilitySigned<br>Tspi_TPM_KillMaintenanceFeature<br>Tspi_TPM_LoadMaintenancePubKey<br>Tspi_TPM_CheckMaintenancePubKey<br>Tspi_TPM_GetRandom<br>Tspi_TPM_StirRandom<br>Tspi_TPM_AuthorizeMigrationTicket<br>Tspi_TPM_GetEvent<br>Tspi_TPM_GetEvents<br>Tspi_TPM_GetEventLog<br>Tspi_TPM_Quote<br>Tspi_TPM_PcrExtend<br>Tspi_TPM_PcrRead<br>Tspi_TPM_DirWrite<br>Tspi_TPM_DirRead | These methods facilitate management of the TPM and platform configuration measurement and reporting. |



IPR2025-01572
Apple EX1048 Page 45

| Component | Area | Command Name | Description |
|---|---|---|---|
| | Key Class Methods | Tspi_Key_LoadKey<br>Tspi_Key_GetPubKey<br>Tspi_Key_CertifyKey<br>Tspi_Key_CreateKey<br>Tspi_Key_WrapKey<br>Tspi_Key_CreateMigrationBlob<br>Tspi_Key_ConvertMigrationBlob | These methods facilitate key management operations performed by the TPM. |
| | Hash Class Methods | Tspi_Hash_Sign<br>Tspi_Hash_VerifySignature<br>Tspi_Hash_SetHashValue<br>Tspi_Hash_GetHashValue<br>Tspi_Hash_UpdateHashValue | These methods are used for general purpose manipulation of message digests and signatures. |
| | Data Class Methods | Tspi_Data_Bind<br>Tspi_Data_Unbind<br>Tspi_Data_Seal<br>Tspi_Data_Unseal | These methods are used to send/receive data where the TPM is the endpoint of communication. |
| | PCR Class Methods | Tspi_PcrComposite_SelectPcrIndex<br>Tspi_PcrComosite_SetPcrValue<br>Tspi_PcrComposite_GetPcrValue | These methods are used to manipulate PCR registers maintained within the TPM. |
| | Callback Functions | Tspip_CallbackHMACAuth<br>Tspip_CallbackXorEnc<br>Tspip_CallbackTakeOwnership<br>Tspip_CallbackChangeAuthAsym<br>Tspicb_CollateIdentity<br>Tspicb_ActivateIdentity | These callback functions are used by TSPI policy objects when caller preferences dictate different or dynamic behavior. |

Figure 4:r - TSP Objects and Methods Summary

# 5. TCG Model for Security Evaluation

TCG recognizes the need for both a technical standard and a process for insuring the standard is properly applied. This section discusses the approach TCG takes to facilitate proper application of the standard. Three interrelated concepts provide a framework. *Evaluation* exposes development and manufacturing processes and products to reviewers. Reviewers provide *certification* that evaluation meets the intended goals of customers, which is to increase assurance of TCG standards compliance. *Accreditation* allows customers to accept certified products into their computing environments and may implement policies requiring certification.

## 5.1 The Context for Evaluation

For evaluation to be meaningful, the right features must be evaluated. Figure 5:a shows the general context of security which motivates the class of features chosen by TCG for definition and standardization. The model captures the objectives of two classes of activity, Owners and Threat Agents. It is evident that the interests' of each are contradictory.

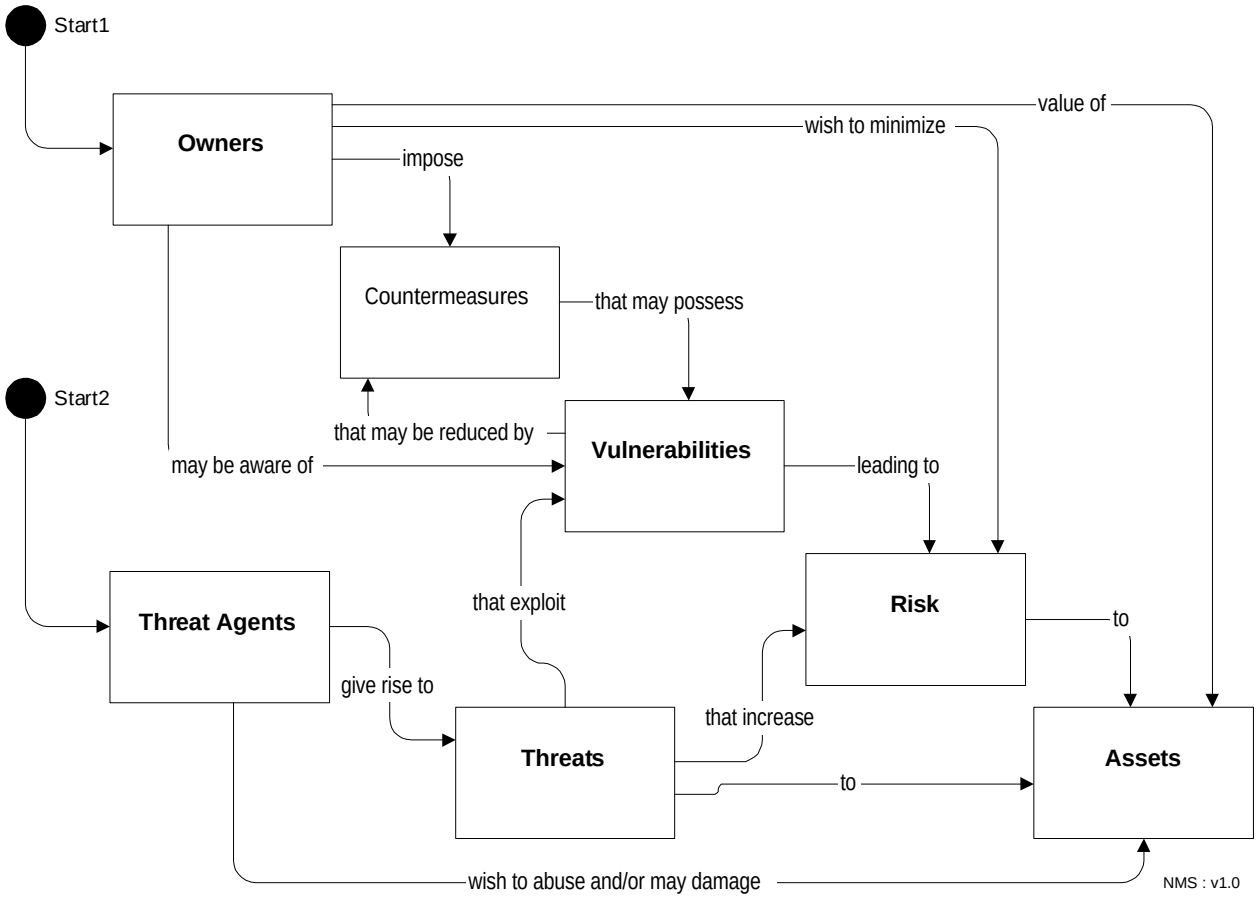



Figure 5:a - ISO 15408 General Security Context

Evaluation supports Owner interests helping to improve countermeasures and become aware of vulnerabilities. TCG accomplishes this in several ways:

- By establishing an Owner of the computing asset – namely the platform.
- By enabling Owner control over the platform.
- By providing measurement, storage and reporting countermeasures that minimize and circumvent threats.

- By requiring evaluation that may be used to identify vulnerabilities.
- By enabling product certification, which in turn can minimize risk.

The TCG feels the general security context is universal to all asset owners. Though the players and their roles may be multiplied and varied, evaluation takes the owners interest to heart. Hence, evaluation is intended to be a collaborative process capturing customer, user and supplier input.

Through collaborative efforts with the TCG Conformance Workgroup, TPM vendors, platform vendors and others protection profiles are defined. *Targets of evaluation (TOE)* may also be defined. A TOE identifies features and functions of the components being evaluated. Security profiles used by evaluators are also defined by TCG working groups. TCG further aids in the evaluation collaboration by coordinating evaluation activities and publishing (where appropriate) evaluation results.

TCG anticipates two targets of evaluation will be defined, one for TPM devices and another for the TBB sub-system. The TPM evaluation provides the assurance that the TPM will work properly and the TBB provides assurance that the TPM is properly connected to the platform. The TPM profile is generic and covers all TPM no matter which platform the TPM resides on. The TBB is platform specific as it deals with platform issues. The combination of these two evaluations allows a consumer of a platform to have an assurance that the system will provide the security and functionality that the platform is claiming to provide.

## 5.2 Goal of Evaluation

The goal of evaluation is to improve assurances that the countermeasures can be trusted to reduce the risks to the protected assets. Evaluation results are a statement that assigns an assurance rating of the countermeasures, assurance being that property of the countermeasures that gives grounds for confidence in their proper operation.

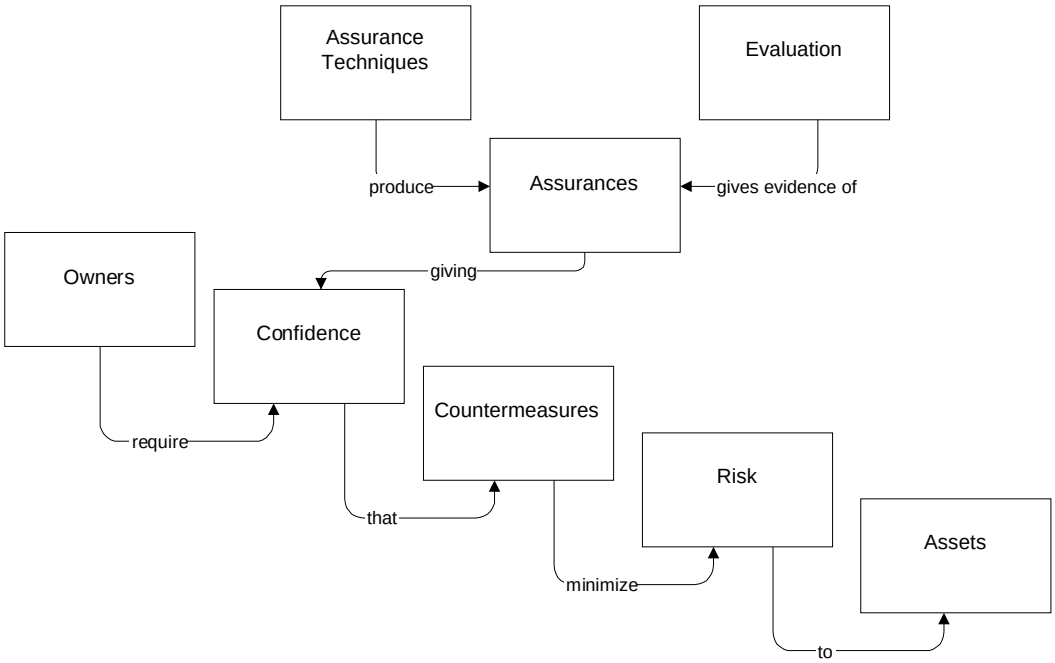



Figure 5:b – ISO 15408 Assurance Context for Evaluation

The context for establishing assurances is depicted in Figure 5:b. It indicates that evaluation is a mechanism through which the platform owner may better trust the platform.

Evaluation is not the only mechanism for owner garnered trust however. Other approaches include user identification and authentication, auditing of user activities, review and control of platform configuration



IPR2025-01572
Apple EX1048 Page 48

changes and monitoring of runtime state. However, these all depend upon the existence of a kernel of computing that stands the test of time. Evaluation is the first step.

## 5.3 The Evaluation Process

The TCG leverages the ISO-15408 (A.K.A. Common Criteria) standard for evaluating computing systems. The criteria serve as a guide for evaluators. It provides common terminology, classification and ratings such that evaluation results may be interpreted and comprehended by a broad audience.

The TCG anticipates participation from many organizations in evaluating and certifying TCG products. ISO-15408 is broadly accepted globally by IT, IS and governmental procurement bodies.

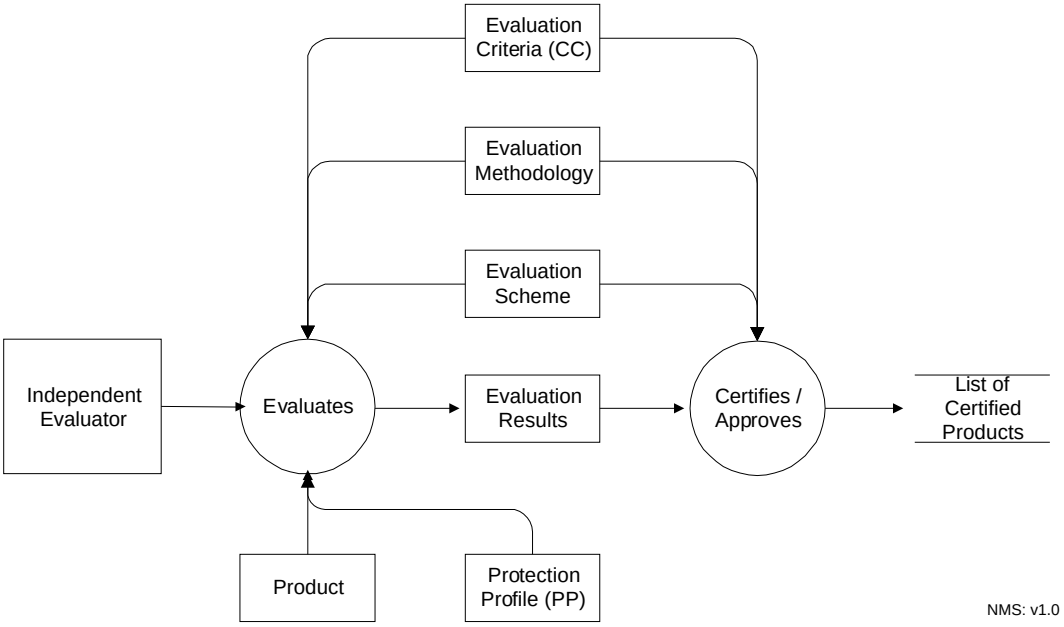



Figure 5:c - ISO15408 Evaluation Context with Process Flow.

### 5.3.1 Inputs to Evaluation

Several factors contribute to a successful evaluation. Figure 5:c depicts the various inputs to the *Evaluates* bubble.

#### *5.3.1.1 Common Criteria*

The CC is presented as a set of distinct but related parts as identified below.

a) **Part 1, Introduction and general model,** is the introduction to the CC. It defines general concepts and principles of IT security evaluation and presents a general model of evaluation. Part 1 also presents constructs for expressing IT security objectives, for selecting and defining IT security requirements, and for writing high-level specifications for products and systems.

b) **Part 2, Security functional requirements,** establishes a set of functional components as a standard way of expressing the functional requirements for TOEs. Part 2 catalogues the set of functional components, families, and classes. Security functionality is divided into 11 classes:

- FAU – Auditing Security
- FCO – Communication
- FCS – Cryptographic Support
- FDP – User Data Protection

- FIA – Identification and Authentication
- FMT – Security Management
- FPR – Privacy
- FPT – Protection of the TOE Security Functions
- FRU – Resource Utilization
- FTA – TOE Access
- FTP – Trusted Path / Channels

Functionality is graded based on a qualification expressing the minimum efforts assumed necessary to defeat its expected security behavior by directly attacking its underlying security mechanisms. The qualification is called Strength of Function (SOF) and is expressed in three grades.

- SOF-basic — A level of the TOE strength of function where analysis shows that the function provides adequate protection against casual breach of TOE security by attackers possessing a low attack potential.
- SOF-medium — A level of the TOE strength of function where analysis shows that the function provides adequate protection against straight forward or intentional breach of TOE security by attackers possessing a moderate attack potential.
- SOF-high — A level of the TOE strength of function where analysis shows that the function provides adequate protection against deliberately planned or organized breach of TOE security by attackers possessing a high attack potential.

TCG recommends a Strength of Function target of **MEDIUM**. However, the TPM and TBB specific profiles may specify an alternate SOF rating. While most elements of TCG meet SOF-medium a few do not. Early versions of TPM and TBB evaluations are expected to receive **SOF-basic** classification.

c) **Part 3, Security assurance requirements,** establishes a set of assurance components as a standard way of expressing the assurance requirements for TOEs. Part 3 catalogues the set of assurance components, families and classes. Part 3 also defines evaluation criteria for Protection Profiles (PP) and Security Targets (ST) and presents evaluation assurance levels that define the predefined CC scale for rating assurance for TOEs, which is called the Evaluation Assurance Levels (EALs).

There are 7 EALs defined. Each level represents increasing degree of effort and complexity:

- EAL1 – Functional testing
- EAL2 – Structural testing
- EAL3 – Methodological testing and checking
- EAL4 – Methodological design, testing and review
- EAL5 – Semi-formal design and testing
- EAL6 – Semi-formal verification of design and tests
- EAL7 – Formal verification of design and tests

Platform vendors determine an EAL that best suites the needs of their customers. The TCG sets a lower limit on EAL to ensure minimum assurances are achieved.

In support of the three parts of the CC listed above, it is anticipated other types of documents will be published, including technical rationale material and guidance documents.

#### *5.3.1.2 Methodology*

Definition of evaluation methodology is reserved primarily for evaluation teams. TCG envisages TPM and platform vendors will contribute heavily to evaluation efforts by leveraging existing testing and validation efforts to achieve a minimum level of assurance.

#### 5.3.1.3 Scheme

The scheme refers to an evaluation authority that sets the standard for evaluations and oversees quality of evaluations. The TCG organization provides the scheme for TCG compliant products.

#### 5.3.1.4 Product

The product is the manufactured part that implements security functionality. Products are identified by manufacturer, model number and revision level.

#### 5.3.1.5 Protection Profiles

Protection profiles provide detailed description of the target of evaluation (TOE) and the goals of evaluation. A cross-section of CC functional area with TOE functionality is expressed in implementation independent style. Anticipated threats, security policies and security objectives are described. Additionally, the target rating sought is identified, one for functionality and another for assurance level.

The Common Criteria (CC) Version 2.1 (ISO/IEC 15408) and Common Methodology for Information Technology Security Evaluation (CEM) 99/008 may be helpful resources for profile writers.

Two profiles are in use by TCG, one for the TPM and another for the TBB. The TPM PP (validation report number CCEVS-VR-02-0022) defines TPM security properties and the TBB PP (validation report number currently unavailable) defines TBB security properties.

#### 5.3.1.6 Evaluators

Evaluators accept all the inputs to evaluation, perform the evaluation and produce evaluation results. Evaluators should include experts knowledgeable in all aspects of the TOE and relevant disciplines.

### 5.3.2 Evaluation Results

Evaluation results contain observation and justification supporting the evaluation team's recommended rating. Evaluation results are organized and formatted following guidelines established by the evaluation *Scheme*. This improves results comparability.

Evaluation results are reviewed by a certifying body to make the final approval decision. Certification procedures are distinct from evaluation procedures.

TCG envisages there will be two products of evaluation, one for the TPM component and a second for the platform containing the TPM.

## 5.4 Certification

The certification process is the independent inspection of the results of the evaluation leading to the production of the final certificate or approval. The certificate is normally publicly available. For certification to be meaningful, evaluation must be performed by competent evaluators. Certification identifies the final rating and provides proof of evaluation. It is noted that the certification process is a means of gaining greater consistency in the application of security criteria.

### 5.4.1 Certified Products List

Certified products are normally entered into a list and made available for public consumption. Publication aids in improving security by raising awareness which in turn may influence procurement.

Certified products lists may also be published electronically for human and automated consumption.

It is anticipated that certification results will be signed taking the form of TCG *Conformance Credentials* .

### 5.4.2 Where Does Certification Authority Originate?

Anybody may assume authority to certify. The objective of certification is to provide credible reference for accreditation, hence customers of certified products determine which organizations are credible.



IPR2025-01572
Apple EX1048 Page 51

TCG feels credibility may be found among many organizations from ranging from product manufacturers / vendors, product consumers and consultants. The product owner ultimately decides which certifier best contributes to assurance and risk management calculations.

TCG hopes that certifying organizations will become valued and integral resources contributing to purchase decisions of trusted products.

## 5.5 Accreditation

Accreditation is the process a customer engages in when determining which technology best supports the customer security and safety objectives. Security policies, practices, guidelines and procedures that capture a safety or security objective are customer specific and influence product procurement decisions.

For accreditation to be meaningful TPM and TBB evaluations must be performed by trusted certifiers. Security policies should define who is trusted and with what responsibility.

### 5.5.1 Protection Profile as Security Policy

Protection profiles contain the security policies implemented in the product. Accreditation agents, whether manual or automated, should perform policy mapping exercises. The protection profile may be helpful when doing the mapping.

### 5.5.2 Site-Specific Security Policy

Accreditation involves determining how best to configure a product to implement customer-specific and site-specific guidelines and practices. *Validation Credentials* contain measurements of discrete components that may be combined to create configurations suitable to accreditation agents.

### 5.5.3 Accreditation and Attestation

Accreditation is not associated exclusively with purchasing and procurement processes. It is often the case that business processes and ad-hoc collaboration dynamically share computing resources with partners having decentralized and disjoint policy setting bodies. Hence, security policies cannot be uniformly enforced by procurement and up-front accreditation.

TCG *Measurement* and *Attestation* provide mechanisms supporting dynamic accreditation. An accreditation agent, programmed with site-security policies may utilize attestation to query the configuration of a target system. Comparing results – including proofs of TPM existence, integrity and certification – the accreditation agent can determine if subsequent computational goals are suitable ala the target system.

This approach to accreditation allows both automation and dynamism to permeate the computing landscape, while ensuring goals of accreditation continue to be enforced.

## 5.6 TCG Specification Conformance

While ISO-15408 evaluation processes will exercise many of the TCG defined standards, it is not a replacement for standards conformance. TCG anticipates standards-compliant implementations of TCG standards will be functionally tested for compliance and completeness.

# 6. Manufacturing & Support Implications of TPM

This section highlights key implications pertaining to the manufacture, deployment and maintenance of TPM devices. These observations may be gleaned through careful reading of this and other TCG documents, but is included here for convenience.

## 6.1 Tamper-resistant Packaging

A tenet of TCG security assurance is the TPM device (whether it be implemented in hardware or software) must be tramper-resistant and not trivially removable or replaceable. This requirement impacts TPM packaging design and TPM-based platforms manufacturing process.

TPM packaging must limit pin probing and EMR scanning. The TPM must be “glued” to the motherboard such that removal procedures themselves are a deterrent and removal of a TMP device is evident to visual inspection.

## 6.2 Field-Upgrade

If the TPM needs to be upgraded after field deployment, the TPM_FieldUpgrade command is used.. Upgrade procedures may require field service personnel to be physically present or have owner authorization to perform upgrade procedures. The TPM_FieldUpgrade() interface is vendor-specific.

## 6.3 International Import/Export of Cryptography

The TPM contains strong cryptography. TCG has attempted to limit access and prevent high-bandwidth use of cryptography. TCG believes TPM devices will not require governmental Export or Import controls. However, it may be necessary for vendors of these devices to obtain waivers.

## 6.4 Key Management Infrastructure

The TCG model for establishing trust in TCG technology may require manufacturers to augment manufacturing processes. Services to create and maintain a database of records, one record for each part manufactured may be needed. Records need to be delivered to end customers through some means to be determined by the manufacturer.

Manufacturers may need to establish public-key signing facilities suitable for signing records in low and high volumes. Some records may contain privacy sensitive information, in which it may be prudent for the manufacturer to protect.

Keys used to sign records should be protected from unauthorized use or disclosure through reasonable but credible IT procedures.

# 7. Glossary

Please refer to the "TCG Design Principles and Glossary".



IPR2025-01572
Apple EX1048 Page 54