

Chapter 7

Introduction to the TPM

Allan Tomlinson

Abstract The Trusted Platform Module (TPM) and smart card devices have many features in common. Both are low cost, tamper resistant, small footprint devices used to provide the basis of a secure computing environment. This chapter presents an introduction to the security mechanisms provided by the TPM highlighting those not typically found on a smart card. The concept of “ownership” is one of the major differences between the TPM and a smart card and this is described in some detail before concluding with a review of some of the security services uniquely provided by the TPM.

Key words: Smart Cards, Trusted Computing, TPM, Security

7.1 Introduction

Smart cards provide a wide range of functionality from simple storage media to complex processors. A common design goal across this diversity however, is the provision of some degree of secure processing, implemented in secure hardware. The Trusted Platform Module, or TPM, is similar to a smart card device in that it is a small footprint low cost security module typically implemented as a tamper resistant integrated circuit (IC). The TPM however, has been specifically designed to be a building block for trusted computing. So although there are many similarities, the different design goals have resulted in a number of differences between the two types of device. One major difference is that the TPM is considered to be a fixed token bound to a specific platform, whereas a smart card is a portable token traditionally associated with a specific user across multiple systems. This is not to

Allan Tomlinson,
Information Security Group,
Royal Holloway, University of London,
e-mail: Allan.Tomlinson@rhul.ac.uk

say that the two technologies are mutually exclusive, but rather that they may be considered as complementary.

The details of smart card architectures and applications are discussed elsewhere in this book so this chapter will focus on the TPM and expand on the “complementary” aspects of this device, describing the specific mechanisms where the TPM differs from a typical smart card IC. In order to fully understand the rationale behind these mechanisms it is important to understand the underlying design goals of trusted computing that guided the development of the TPM.

This chapter therefore begins with some background on trusted computing to explain the features that the TPM is designed to support and the rationale behind the design decisions that were made. The fundamental features of the TPM that emerged from these design constraints are described in section 7.3. This section focuses on the functionality of the basic TPM building blocks to expose some of the differences between the design of smart card ICs and the TPM. Many of the differences that appear, arise from the contrasting requirements for ownership and management between the two types of device. Section 7.3 therefore looks at these concepts in some detail. The differences between the two types of device result in differences in functionality, and section 7.4 looks at some of the security services not found in typical smart card devices, which are fundamental to the operation of the TPM.

7.2 Trusted Platforms

Where smart cards may be considered as general purpose security processors, the TPM has been designed specifically to support trusted computing platforms. Therefore, in order to understand the TPM design requirements, it is first necessary to understand what the desirable features of a trusted platform are. To do this, a definition is required as to exactly what is meant by the term “trusted platform”.

The concepts of trusted computing, and a Trusted Computing Base, or TCB, are not new and are described in many publications ranging from the Orange Book [3] through to more recent material that describe these ideas within the context of contemporary developments in computer security [2, 5, 15]. One such development is the emergence of the Trusted Computing Group¹ (TCG) that has attempted to define what is meant by a trusted computing platform and that has produced a series of standards that can be used to design trusted platforms. The TCG defines trust to be “the expectation that a device will behave in a particular manner for a specific purpose” [16]. The TCG’s definition of a trusted platform therefore, is a platform that behaves in such a manner. This is, necessarily, a rather high level definition: but what this means is that any entity that interacts with such a trusted platform can be given some degree of assurance that the platform (system or computer) will behave in the way that entity expects it to. Providing assurance of expected behaviour does not in

¹ <https://www.trustedcomputinggroup.org>

itself provide any security. To achieve that, the entity relying on this assurance still has to ascertain that the “expected behaviour” is indeed secure. However, assuming that the relying entity is satisfied that the expected behaviour is secure then he may, for example, be assured that any data given to the system is kept confidential, or that no malware is running on the platform.

There are many ways to design trusted computing platforms that allow statements about expected behaviour to be made [15], but the approach taken by the TCG is to have some physically secure trusted component that can be used as a foundation upon which trust in the rest of the system can be built. The purpose of the TPM is to provide this foundation. For this chapter, a more appropriate definition of a trusted platform is provided by Pearson [13] who states that:

“A Trusted Platform is a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes”

or Balacheff et al. [1] who say:

“A trusted platform (TP) is defined as a computing platform that has a trusted component, which is used to create a foundation of trust for software processes.”

It is perhaps appropriate at this point to make a subtle distinction between what is meant by a *trusted component*, such as the TPM, and a *trustworthy component*. One generally accepted definition of these terms is given by Anderson [9, 10] who states that:

“The proper definition is that a *trusted* system or component is one whose failure can break the security policy, while a *trustworthy* system or component is one that won't fail”

By implementing this trusted component, the TPM, as a tamper proof IC; and binding it to the platform, usually on a printed circuit board containing a more powerful processor capable of running software applications; the TPM can be used as the foundation of trust for higher level processes that run on the main processor.

7.2.1 Fundamental Features of a Trusted Platform

In order to establish this foundation of trust, the TPM is expected to provide a fundamental set of security features which have been defined by the TCG. The minimum set of features that a trusted platform should have are: protected capabilities; integrity measurement; and integrity reporting [16]. Providing support for these features leads to the definition of the security requirements of the TPM.

Protected Capabilities

To meet the requirements of a trusted platform, according to the TCG [16], the system should provide some form of protected capabilities. In the TPM design prin-

ciples specification, the concept of protected capabilities is used to “distinguish platform capabilities that must be trustworthy” [17].

These trustworthy protected capabilities are abilities to execute a command or set of commands on the TPM which access *shielded locations* where it is safe to operate on sensitive data [16]. Examples of protected capabilities in the TPM include protection and reporting of integrity measurements (described below); and storage and management of cryptographic keys.

Integrity Measurement and Storage

Any trusted platform will need some means of measuring how it is configured and what processes are running on that platform. The increasing complexity of personal computers and software applications has resulted in an increase in the number of processes that run on a typical PC. Many of these processes are launched implicitly, rather than explicitly by the user. Under such circumstances it is difficult to tell if the code being executed on a particular platform is a legitimate process or not, and consequently, if this particular platform can be trusted. It is important therefore, if a platform is to be trusted, that it has some means of measuring the integrity of the processes it is running. This measurement should result in some form of *integrity metric* [16] which Pearson defines as “a condensed value of integrity measurements” [14]. This integrity metric can then be compared with acceptable values for a trusted platform. Having obtained an integrity metric for the platform it is often useful to store that value somewhere for later use. Of course such data needs to be held in secure storage locations such as would be provided by the protected capabilities described above.

In measuring platform integrity in this manner there has to be a starting point. It may be acceptable for a particular application to perform integrity measurements provided that the application itself is trustworthy. Even if the operating system can verify the integrity of this application, the integrity of the operating system too has to be verified. Ultimately there will be some integrity measurement process that exists which cannot itself be verified. This process is known as the Root of Trust for Measurement, or RTM. This is the starting point in the chain of integrity measurements, and ideally this process should run on tamper proof hardware, with the execution code being stored in secure storage.

Integrity Reporting

The third requirement for a trusted platform [16] is that it should be able to report its configuration to a challenger who requires this information in order to decide how much trust to place in the platform. In other words, the platform should have some means to report its integrity metrics and to vouch for the accuracy of this information. This attestation requires the generation of evidence that the challenger can rely on in making its trust decision. The implication here is that the integrity

metrics can be signed by the trusted platform and that the challenger has a certificate that can be used to verify the signature.

7.2.2 Additional Features

While the above features would allow the implementation of a basic trusted platform as defined by the TCG, there are other features that would be required to create a more flexible implementation of a trusted platform. These include: confidentiality and integrity protection; secure storage; and process isolation.

Confidentiality and Integrity Protection

In addition to the protection of integrity metrics, a trusted platform should provide both confidentiality and integrity protection to any data as required by the user. This could include both user data, and application code. Moreover, these security services should be available to protect this data while it is being stored, and during the execution of any process.

Secure Storage

The provision of confidentiality requires that a trusted platform is able to encrypt any user data for secure storage. Access to the data is controlled by a securely stored cryptographic key. Similar to encryption is the concept of sealing. In this case access to the data is controlled by platform state, the use of a cryptographic key is optional. This means that data can be sealed to a set of integrity metrics that reflect the platform state. The data will then only be accessible when the platform is in the desired configuration. The sealing mechanism can be used, for example, to ensure that no rogue applications are running before access is granted to sensitive data.

Process Isolation

The secure storage and integrity checking mechanisms will protect data during storage. To protect data during execution the provision of process isolation is necessary. The concept here is to provide an isolation kernel between the hardware and any operating system. This isolation kernel is used to create and manage multiple secure compartments. These compartments exist in parallel, on the same machine, and access is managed exclusively by the isolation kernel. Each compartment can then run its own operating system and applications in isolation from any other processes that are executing in parallel. In this way, application code and data can be protected during execution. Furthermore, the use of an isolation kernel greatly sim-

plifies the validation of acceptable integrity metrics: by isolating processes, the set of acceptable platform configurations can be reduced to one operating system and one application only.

7.3 TPM Features

The previous section described the main security requirements for the TPM and the rationale behind these requirements. These high level requirements for a trusted platform provided some of the input to the development of a series of standards which are published by the TCG, including the specification of the TPM. This specification is written as a collection of three documents, currently at version 1.2. The documents describe the TPM design principals [17], which will be reviewed in more detail in the following section; and the data structures [18] and commands [19] used to control the TPM. A discussion of TPM structures and commands is beyond the scope of this chapter but may be found in other work [4, 12].

7.3.1 TPM Components

When comparing the TPM to smart cards, an important point to note is that the TCG specifications do not mandate that the TPM is implemented as an IC. The standards define TPM *functionality*, leaving it open for developers to implement this functionality as they wish, either in hardware or software. Having said that, most commercially available implementations of the TPM are hardware based and produced by the major IC manufacturers. The main building blocks of the TPM are shown in figure 7.1.

The TPM itself, however implemented, must be physically protected from tampering. In PCs, for example, this may be accomplished by physically binding the TPM to the motherboard to make it difficult to disassemble or transfer to a different platform. This is not a requirement for the TPM to be tamper proof, but tamper resistant. The TCG also require that tamper evidence measures should be deployed to enable detection of any tampering by physical inspection.

It is important to note when looking at figure 7.1, that the TPM standards do not specify the communications interfaces or bus architecture, leaving these decisions to be made by the developers.

7.3.2 I/O Block

The TCG do, however, specify an interface serialisation transformation that allows data to be transported over virtually any bus or interconnect. This is one of the

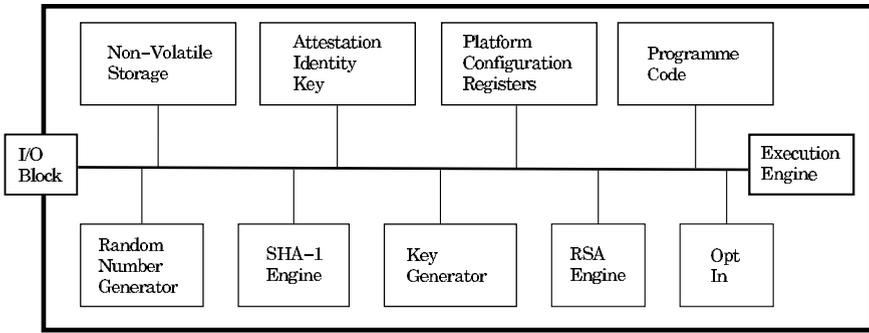


Fig. 7.1 TPM Building Blocks

functions of the I/O block. This block manages information flow between the components illustrated in Fig. 7.1, and between the TPM and the external bus. Since the I/O block is managing data flow, it is also able to control access to the various TPM components. The access rights are determined by flags maintained by the Opt-In block.

7.3.3 Non-Volatile Storage

As with a smart card IC, a TPM has some non-volatile memory to store long term keys. Two long term keys are stored in non-volatile memory on the TPM. The first of these is the Endorsement Key (EK); the second key is the Storage Root Key (SRK) which forms the basis of a key hierarchy that manages secure storage.

The TPM also uses non-volatile memory to store *owner* authorisation data. This authorisation data is, in effect, the owner’s password and is set, not by the manufacturer, but during the process of taking ownership of the TPM. There are also a number of persistent flags related to access control and the Opt-In mechanism, discussed in section 7.3.12, that need to be stored in non-volatile memory.

Endorsement Keys

The Endorsement Key (EK) is a fundamental component of a TPM. It is also something unique to the TPM which is not found on a smart card device. It is therefore important to consider how the TPM uses this key in a little more detail.

For the TPM to operate, it must have an endorsement key embedded in it. To be more precise, it must have an endorsement key pair, of which the private key is embedded in the TPM and *never leaves* it. The public EK is contained in a certificate and is only used in a limited number of procedures. The reason for limiting the use of the EK certificate is because the EK is unique to each TPM and consequently may

be used to identify the device, and by extension the platform. Therefore, to protect user privacy when interacting with other entities, the use of the EK is restricted and internally generated aliases, the Attestation Identity Keys, or AIKs, are used for routine transactions.

TPM manufacturers will provide the endorsement key pair and store this in tamper resistant non-volatile memory before shipping the TPM. A certificate, or *endorsement credential*, can then be created which contains the public EK and information about the security properties of the TPM. This endorsement credential should be signed by a certification authority, known as the TPME or Trusted Platform Module Entity, who can attest to the fact that the key contained in the certificate is a public EK whose corresponding private EK is stored in a TPM that conforms to the TCG standards. This TPME may be a third party or, if authorised to do so, it may be the manufacturer themselves.

Some organisations who wish to use the TPM may, for security reasons, prefer to use their own endorsement keys. To accommodate this, the standards allow the EK to be deleted and re-installed by the user. Of course if the user generated endorsement credential is not signed by a TPME, its use may be limited.

The purpose of the endorsement credential is to prove that the corresponding private EK is stored in a genuine TPM. So, in keeping with policy to control exposure of the public EK, the private EK is *never used to generate signatures*. Thus, the public EK is never required to verify signatures so it does not have to be widely distributed. The public EK is only used for encrypting data sent to the TPM during the process of taking ownership and the process of creating AIK certificates. These processes are described in sections 7.3.12 and 7.4.4. Encrypting data with the public EK ensures that the plaintext can only be recovered by the particular TPM identified in the endorsement credential.

7.3.4 Attestation Identity Keys

As mentioned above the TPM Endorsement Key and Storage Root Key are stored in non-volatile memory. These keys never leave this secure storage during normal operation. A third type of key, the Attestation Identity Key (AIK), may also be stored within the TPM. This key may be regarded as an alias for the Endorsement Key. Each TPM can support many AIKs, thus the user can have many unlinkable keys that can be used to maintain anonymity between different service providers who require proof of identity. These AIKs must, therefore, be persistent and although they could be stored on the TPM non-volatile memory, for practical reasons the standards recommend keeping the AIK keys in secure external storage. The TPM however must provide a volatile storage area where one or more AIK keys can be loaded when in use.

7.3.5 Platform Configuration Registers

The Platform Configuration Registers (PCR) are unique features of the TPM architecture and are used to store *integrity metrics*. The integrity metrics stored in these registers measure the integrity of any code, from BIOS to applications, typically before the code is executed. Platform Configuration Registers may be implemented in volatile or non-volatile storage. However, these registers *must* be reset whenever the system loses power or re-starts. If the registers were not reset then old integrity metrics might remain in the PCRs after a platform is re-booted and reconfigured. The standards specify that a TPM must have at least 16 Platform Configuration Registers and that each register stores 20 bytes. Registers 0 to 7 are reserved for exclusive use by the TPM, the remaining registers are free for use by the operating system and any application.

7.3.6 Programme Code

In common with smart cards, the TPM requires storage for the firmware that is used to initialise the device.

If the programme code is stored permanently on the tamper proof TPM then it would be reasonable to assume that it is trustworthy. Thus there would be no need to check its integrity making this the obvious location to store the code that carries out the integrity checks on all other platform devices and code. That is to say, the programme code on the TPM is the obvious “root of trust” for integrity measurements described in section 7.2. The TCG refer to such a root of trust as the CRTM, or Core Root of Trust for Measurement. Although the TPM programme code is the obvious choice for the CRTM, implementation decisions often require the CRTM be located in other firmware such as the BIOS boot block. Regardless of where the CRTM resides it should be considered as a *trusted component* of the system since if it fails all security policy based on integrity measurements will be broken.

7.3.7 Execution Engine

Like many smart cards, the TPM has an execution engine which runs the programme code described above. The execution engine responds to external commands by selecting the required programme code and executing it on the TPM.

7.3.8 Random Number Generator

Another unique feature of the TPM is the inclusion of a *true* random bit stream generator. Again the implementation is left to the developers so long as some random source is used, rather than a deterministic method. Having a true random bit generator is extremely valuable in any security application. In the TPM, random bit streams are used to seed a random number generator. The random numbers produced by this generator may then be used to construct keys for symmetric cryptographic applications. The random numbers may also be used to provide nonces and, by mixing with user input, to increase the entropy in pass phrases

7.3.9 SHA-1 Engine

The SHA-1 message digest engine is an implementation of the Secure Hash Algorithm [6] SHA-1. This algorithm hashes the input data and produces a 20-byte digest. It also forms the basis of an HMAC [7, 8] (Hash Based Message Authentication Code) engine, and is used in a number of cryptographic procedures carried out by the TPM, for example: in the computation of digital signatures and for creating key objects where a hash of the key may be required as part of an integrity protection mechanism.

7.3.10 RSA Key Generation

Generating keys suitable for use with the RSA algorithm [11] can be a computationally intensive task and since such keys are widely used in the TPM, for signing and providing secure storage, the standard specifies that the TPM should include a module specifically for this task [16]. The standard requires that a TPM is able to support keys *up to* a 2048 bit modulus. Moreover, there are certain keys used with the TPM that must have *at least* a 2048 bit modulus.

In other words, all implementations of the TPM are required to support up to 2048 bit RSA. Some keys are allowed to have a smaller modulus than this, depending on what they are used for. However there are certain keys that must have a 2048 bit modulus - or greater. Of course if the modulus is greater than 2048 bits there is no guarantee that all implementations of the TPM will support this since the only requirement is that the TPM is able to support keys with up to a 2048 bit modulus.

7.3.11 *RSA Engine*

Just as the generation of RSA keys is computationally complex, so is the execution of the algorithm itself. Therefore, the standards also require the TPM to have a dedicated RSA engine used to execute the RSA algorithm. The RSA algorithm is used for signing, encryption, and decryption. Maintaining the principle of key separation, the TPM uses dedicated signing keys for signing data, and separate storage key pairs for encryption and decryption. It is worth noting that the TPM does not mandate the use of any symmetric crypto-engines. According to the TCG [16], “The TCG committee anticipates TPM modules containing an RSA engine will not be subject to import/export restrictions.”

7.3.12 *Opt-In*

The Opt-In component and the concept of ownership, represent one of the biggest differences between smart cards and the TPM. Smart cards are in general, owned and customised by the Issuer before the consumer receives the device. The TCG, however, conscious of the perception that TPM enabled platforms will somehow be controlled by large remote organisations, have been very careful to provide mechanisms to ensure that it is *the user* who takes ownership and configures the TPM. The TCG policy is that the TPM should be shipped “in the state that the customer desires” [16]. Thus, it is up to the user to opt-in to use the TPM. Users are not compelled to use trusted computing, they only opt-in if they choose to do so by taking ownership of the device.

During the process of taking ownership, the TPM will make transitions through a number of states depending on the initial state in which the device was shipped. The state in which the TPM exists is determined by a number of persistent and volatile flags. Changing the state of these flags requires authorization by the TPM owner, if he exists, or demonstration of physical presence. Proving ownership is discussed below, but the means of proving physical presence is determined by the platform manufacturer. The key point is that no *remote* entity, other than the TPM owner, should be able to change the TPM state [17].

The function of the Opt-In component is to provide mechanisms and protection to maintain the TPM state via the state of these flags.

TPM Operational States

There are several mutually exclusive states in which the TPM can exist, ranging from disabled and deactivated through to fully enabled and ready for an owner to take possession [16]. In the *disabled* state, the TPM restricts all operations except the ability to report TPM capabilities and to accept updates to the PCRs. Once a TPM is *enabled*, all its features are made available, provided ownership has been

established. If an owner has not already been established then the transition from disabled to enabled requires proof of physical presence.

A second state variable indicates whether the device is *activated* or *deactivated*. The deactivated state is similar to the disabled state. The difference is that when deactivated, a TPM may still switch between different operational states, for example to change owner or to activate. Once activated, all features are available [16].

A third state variable relates to ownership and determines whether the TPM is *owned* or *unowned*. For a TPM to be owned by a user, it must have an endorsement key pair, and a secret *owner authorisation data* known by the owner. Once in the owned state, the owner of the TPM may perform all operations including operational state change. The TPM needs to have an owner and be enabled for all functions to be available [17].

Depending on the configuration, cases may arise where different states have overlapping influence. As explained in the TCG architectural overview specification [16], in such situations, where TPM commands are available by one mode and unavailable by another mode, precedence is given to making the command unavailable.

Taking Ownership

When a user takes ownership of a TPM they establish a shared secret, referred to as *owner authorisation data*, and insert this data into secure storage on the TPM. This is, in effect, the owner's password and being able to demonstrate knowledge of this secret provides proof of ownership. Once owned, the TPM can control access to certain protected operations by requiring proof of ownership which can be demonstrated by entering the *owner authorisation data* to authenticate the owner.

The process of taking ownership requires that the owner authorisation data is protected from eavesdropping or theft by a malicious third party. This is one case where the endorsement credential is used, since the EK is the only key an un-owned TPM has. It is the EK that establishes the secure channel to transmit the authorisation data to a genuine TPM. So, during the process of taking ownership, the owner requests the endorsement credential and after verifying this credential, retrieves the public EK and uses this to encrypt the shared secret. Only the TPM identified in the endorsement credential has access to the private EK, therefore the shared secret is only made available to the intended TPM. The process of taking ownership is completed by the creation of the Storage Root Key (SRK) which is generated by the TPM and stored in non volatile memory. The SRK forms the root of a key hierarchy used to provide secure storage and, as with the private EK, the SRK *never leaves* the TPM.

7.3.13 Other Features

The TPM provides several other useful security features that are not always available in smart cards. One of these is the provision of monotonic counters which can, for example, provide a secure mechanism to prevent replay attacks. A second feature is the provision of time-stamping, although it is important to note that no absolute measure of time is possible, only measurement of time intervals. Finally, the TPM should provides mechanisms to create and manage audit trails

7.4 TPM Services

To conclude this chapter, this section presents an overview of the TPM operation to illustrate the basic security services provided. The foundation for the provision of these services is the concept of a “root of trust” from which other services such as authenticated boot, secure storage, and attestation can be constructed.

7.4.1 Roots of Trust

As defined in section 7.2, a trusted platform has at its foundation some trusted component. Since all other trust is built upon this foundation it is known as a *root of trust*. An analogy may be made with root certification authorities (CA) in a public key infrastructure (PKI). Although users may not trust the public key presented in a certificate, or even the authority who signed the certificate, so long as there is someone at the top of the certificate chain whom the user trusts, then it is reasonable to trust the key - assuming of course that all signatures are verified. The entity at the top of the certificate chain is often referred to as the *root CA* and his certificate referred to as a *root certificate*. In most web browsers, many root certificates are installed with the application and trusted implicitly by the users. Failure of this component, the root certificate, breaks the security policy of the PKI.

In the trusted platform architecture defined by the TCG, there are three distinct roots of trust: a Root of Trust for Measurement (RTM), a Root of Trust for Storage (RTS) and a Root of Trust for Reporting (RTR).

The Root of Trust for Measurement (RTM) must be trusted to generate integrity measurements for the processes that are running on the platform. This should ideally be a tamper proof component that boots very early in the boot process and therefore is able to measure all other components that are loaded after it. The TPM is an ideal candidate for the CRTM, but in practice the CRTM is more likely to be found in the BIOS boot block. This has been discussed in sections 7.2 and 7.3.6.

The Root of Trust for Storage (RTS) is a trusted component that provides confidentiality and integrity protection. The RTS can then be trusted to store either data, e.g. PCRs, or keys such as the SRK that allow data to be securely stored in external

locations. Finally, the Root of Trust for Reporting (RTR) is a trusted component that provides reports of any integrity measurements that may have been made - attesting to the platform configuration. Both the RTS and RTR are provided by the TPM.

7.4.2 *Boot Process*

Figure 7.2, taken from the TCG Architectural Overview [16], shows the system boot process. The BIOS boot block, or Trusted Boot Block (TBB) as shown in the diagram, contains the CRTM, and is the first process to boot. The CRTM is a trusted component and its integrity is not measured by any external code, but it may perform a self-check of its own integrity. Although not illustrated in the diagram, the CRTM should measure the rest of the BIOS before loading it. Once the BIOS is loaded it takes control and measures the integrity of the OS Loader as shown in step 1 of the diagram. Control then passes to the OS Loader in step 2, and the OS Loader measures the integrity of the operating system. This process continues until the applications are loaded and executed in step 6.

The integrity measurements at each stage are made by creating a SHA-1 digest of the code to be loaded. This digest is stored in one of the PCR registers, which are initialised to zero. The new integrity metric however does not simply overwrite the old PCR value. The process of updating (or extending) the PCR value concatenates the 20 bytes of data already held in the PCR with the 20 bytes of new data calculated by hashing the new code. These 40 bytes of data are then hashed again using the SHA-1 algorithm and the result written to the original PCR. In pseudo code: $PCR \leftarrow hash(PCR \parallel hash(new\ code))$. This way the PCR can store an unlimited number of measurements.

In order to interpret the value contained in the PCR, it is necessary to know the individual digests that have been added to it. These data are stored externally in what the TCG refer to as the Stored Measurement Log. Thus, if the data in the stored measurement log is known, and the PCR values are known, and trusted, then a challenger can verify the state of the platform.

7.4.3 *Secure Storage*

In section 7.3.12 it was mentioned that the process of taking ownership resulted in the creation of a Storage Root Key, or SRK. This key is generated by the TPM and never leaves the device. It can only be accessed by demonstrating knowledge of a shared secret, in this case the *SRK authorisation data*. This shared secret is similar to the *owner authorisation data* and loaded into the TPM at the same time, during the process of taking ownership. As with the *owner authorisation data*, the *SRK authorisation data* is encrypted by the endorsement key before being sent to the TPM.

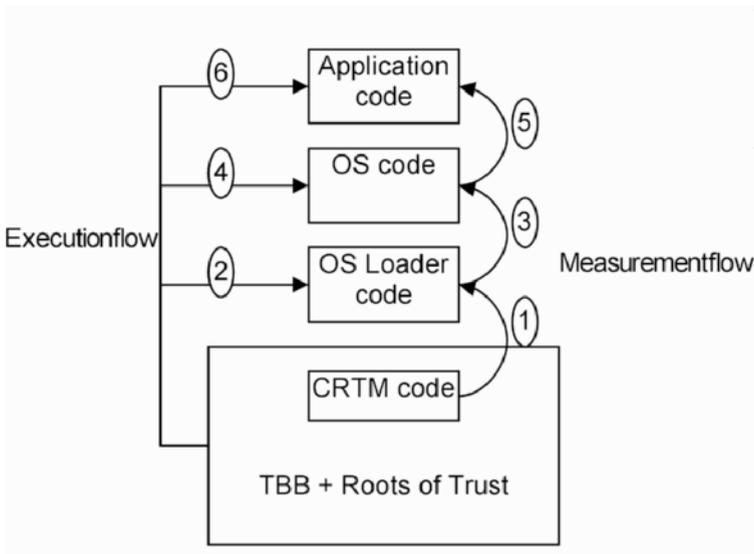


Fig. 7.2 Boot Process [16]

The SRK forms the root of a key hierarchy as illustrated in figure 7.3 which has also been taken from the TCG Architecture Overview [16]. This key hierarchy allows data, or keys, to be encrypted such that they can only be decrypted by accessing the TPM. In the diagram, the opaque data is encrypted under a specific storage key. The storage key is also encrypted and stored external to the TPM. To access this data the encrypted storage key is loaded into the TPM, via the key cache manager, and deciphered by the storage root key. Since the SRK never leaves the TPM, and the TPM is physically bound to the trusted platform, the opaque data can only be deciphered on that platform.

The TPM provides two mechanisms for secure storage: binding and sealing. The *binding* operation encrypts the data using a key that is managed by a particular TPM as described above. The *sealing* process adds to this by only allowing the deciphering process to proceed if the platform is in a specific configuration. This configuration is determined by data held in the PCR registers. Thus, when data is sealed, not only must the same platform be used to unseal the data, but that platform must be in a predetermined configuration before the data can be recovered.

7.4.4 Attestation

The AIK keys shown in figure 7.3 are the Attestation Identity Keys mentioned in section 7.3.1. The purpose of an AIK is to provide user privacy when communicating with different sources. Although EK could be used to secure communications

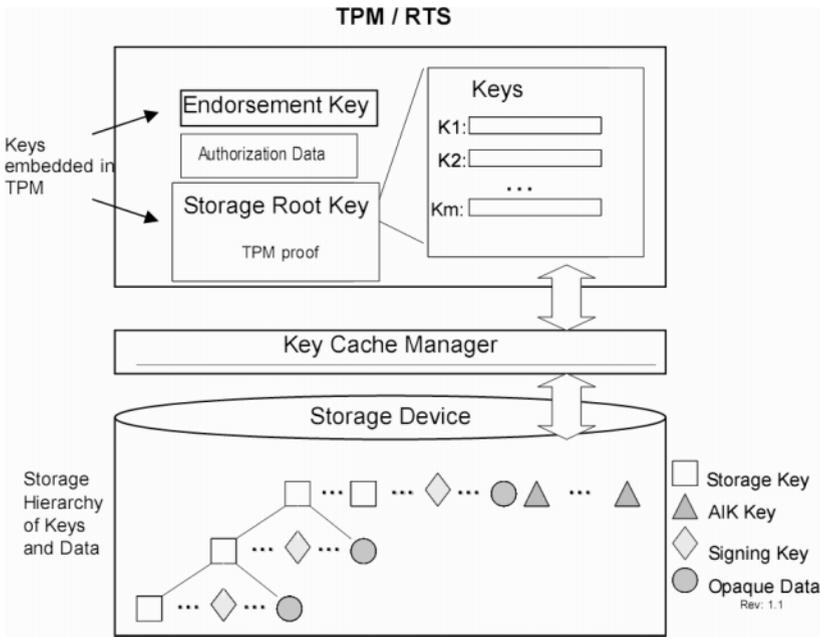


Fig. 7.3 Secure Storage [16]

with different sources, since the EK is unique to the TPM, this could potentially allow the platform’s identity to be linked between each source it chose to communicate with. The idea of the AIK is to provide a unique unlinkable identity for the TPM, for use with each different source. In each case the AIK acts as an alias for the EK. The AIK credential is a certificate containing the AIK public key which proves that the corresponding private key is bound to a genuine TPM. This proof is guaranteed by a signature on the credential created by a trusted third party known as a “privacy CA”.

To obtain an AIK a request is sent to the privacy CA together with the endorsement credential. This is the second case where the public EK is exposed. The endorsement credential proves to the privacy CA that the request came from a genuine TPM. In response to the request, the privacy CA creates and signs the AIK credential, and encrypts this under the public EK contained in the endorsement credential. Thus the AIK is cryptographically bound to the TPM that contains the private EK. The users can create as many AIK keys as they wish and provided the privacy CA is trustworthy, these keys will remain unlinkable and provide the privacy required when communicating with different sources.

The private AIK key is managed by the TPM as illustrated in 7.3 and may be freely used to generate signatures. In particular, the private AIK can sign the contents of the PCR registers. This is how the TPM can be used to attest to the platform configuration. A relying party in possession of a public AIK can now challenge the

trusted platform, and the attestation mechanism can be used to provide an integrity report describing the platform state.

7.5 In Conclusion

This chapter has reviewed the structure of the TPM and identified several unique features that distinguish it from a typical smart card. One of the main differences between the two is the concept of ownership. Where smart card cards are usually owned and customised by the Issuer, the TCG have been careful to ensure that the TPM is owned by the user.

The TPM is designed to provide a number of security services to a platform which differ from those services provided by smart cards. These services include secure storage that can be used to protect data in a hostile environment and, when used with integrity reporting, can ensure that this data is only accessed in a controlled manner. The TPM also provides the mechanisms to implement authenticated boot processes. Perhaps the most unique feature of the TPM is the attestation mechanism. This allows challengers to verify the state of the platform, and which applications are running. It remains, however, the challenger's responsibility based on this attestation, to decide whether or not the platform is actually trustworthy.

References

1. B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler, *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall, 2003.
2. C. Mitchell, ed., *Trusted Computing*. London, UK: IEE Press, 2005.
3. D. of Defense, "DoD 5200.28-STD: Department of defense trusted computer system evaluation criteria", tech. rep., Department of Defense, 1985.
4. E. Gallery, "An overview of trusted computing technology", in *Trusted Computing* (C. J. Mitchell, ed.), IEE, 2005.
5. E. Gallery and C. J. Mitchell, "Trusted mobile platforms", in *FOSAD '07, International School on Foundations of Security Analysis and Design*, vol. 4677 of LNCS, (Bertinoro, Italy), Springer-Verlag, Sep 2007.
6. FIPS, "Specifications for the Secure Hash Standard", Tech. Rep. 180-2, Federal Information Processing Standards, National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161, Aug 2002.
7. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication", RFC 2104, IETF, 1997.
8. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication", in *Advances in Cryptology - Crypto '96* (N. Kobitz, ed.), vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, 1996.
9. R. Anderson, "Cryptography and competition policy: issues with 'trusted computing'", in *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, (New York, NY, USA), pp. 3V10, ACM Press, 2003.
10. R. J. Anderson, *Security Engineering - a Guide to Building Dependable Distributed Systems*. Wiley, 2001.

11. R. L. Rivest, A. Shamir, and L. M. Adelman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, pp. 120V126, 1978. More Information Available via. <http://citeseer.ist.psu.edu/rivest78method.htm>. Cited on 3rd Oct, 2007
12. S. Kinney, *Trusted Platform Module Basics: Using TPM in Embedded Systems*. Elsevier, 2006.
13. S. Pearson, "Trusted computing platforms, the next security solution", Technical Report HPL-2002-221, Hewlett-Packard Laboratories, Nov 2002.
14. S. Pearson. (ed.): "Trusted Computing Platforms: TCPA Technology in Context", Prentice Hall, 2003.
15. S. W. Smith, *Trusted Computing Platforms: Design and Applications*. Springer-Verlag, 2005.
16. TCG, "TCG Specification Architecture Overview", TCG Specification Revision 1.4, The Trusted Computing Group, Portland, OR, USA, Aug 2007.
17. TCG, "TPM Main, Part 1 Design Principles", TCG Specification Version 1.2 Revision 103, The Trusted Computing Group, Portland, OR, USA, Jul 2007.
18. TCG, "TPM Main, Part 2 TPM Data Structures", TCG Specification Version 1.2 Revision 103, The Trusted Computing Group, Portland, OR, USA, Jul 2007.
19. TCG, "TPM Main, Part 3 Commands", TCG Specification Version 1.2 Revision 103, The Trusted Computing Group, Portland, OR, USA, Jul 2007.