TCG Infrastructure WG

# CMC Profile for EK/Platform Certificate Enrollment for TPMv1.2

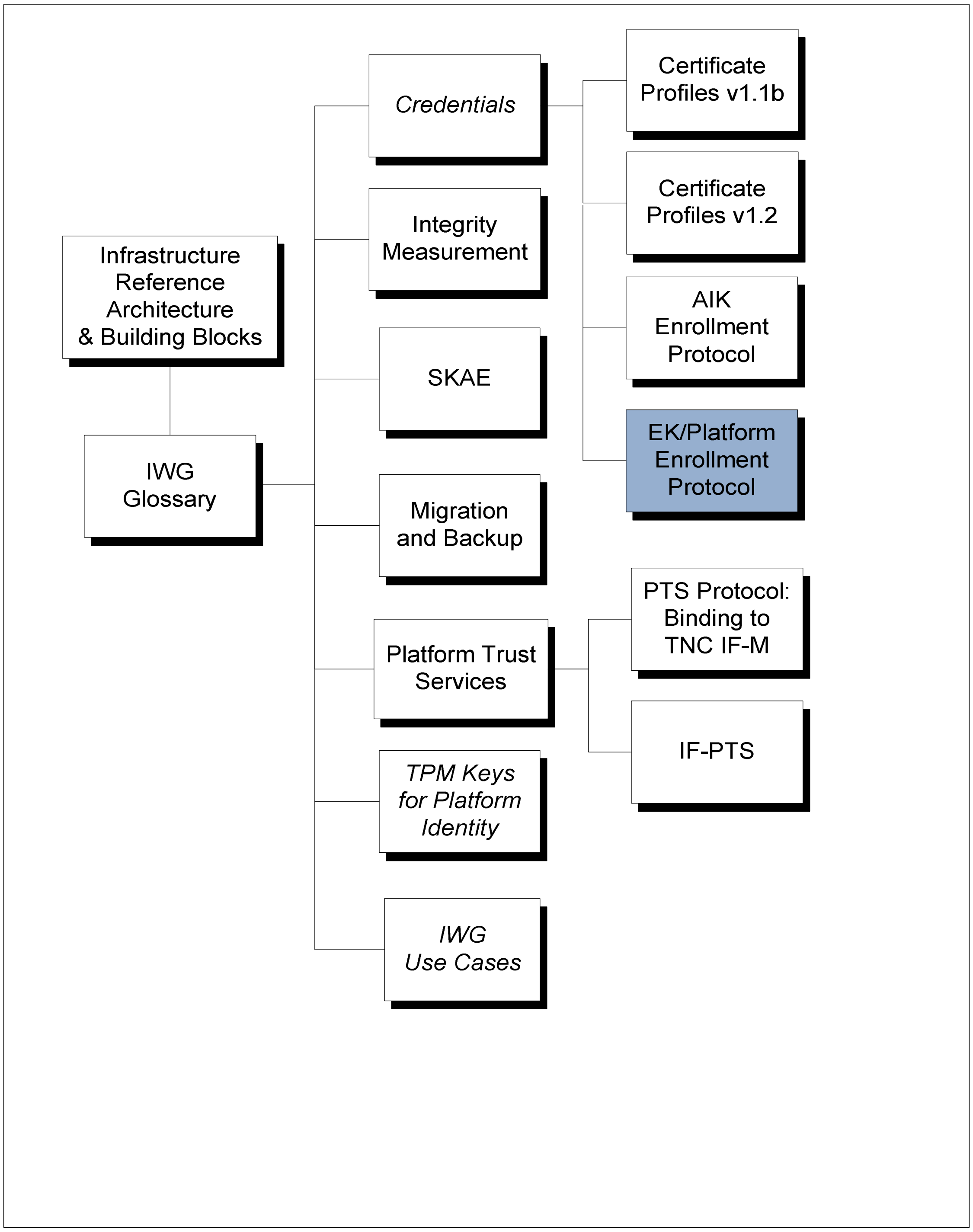
**Specification Version 1.0**
**Revision 5**
**3 April 2013 Published**

**Contact:**
admin@trustedcomputinggroup.org

**TCG PUBLISHED**

TCG

# IWG Document Roadmap


Infrastructure Reference Architecture & Building Blocks
IWG Glossary
Credentials
Integrity Measurement
SKAE
Migration and Backup
Platform Trust Services
TPM Keys for Platform Identity
IWG Use Cases
Certificate Profiles v1.1b
Certificate Profiles v1.2
AIK Enrollment Protocol
EK/Platform Enrollment Protocol
PTS Protocol: Binding to TNC IF-M
IF-PTS

## Acknowledgements

The TCG wishes to thank all those who contributed to this specification. This document builds on numerous works produced by the various working groups in the TCG.

| | |
|---|---|
| Carlin Covey | Freescale |
| Ken Goldman | IBM |
| Wyllys Ingersoll | Sun Microsystems |
| Greg Kazmierczak | Wave Systems |
| Scott Kelly (Editor) | Hyperthought Solutions |
| Carolin Latze (IWG Co-Chair) | Swisscom |
| Max Pritikin (IWG Co-Chair) | Cisco |
| Graeme Proudler | Hewlett-Packard |
| Gloria Serrao | U.S. Government |
| Paul Sangster | Symantec |

## Table of Contents

1 Introduction ........ 6
1.1 Scope and Audience ........ 6
1.2 Keywords ........ 6
1.3 Definitions ........ 6
2 Background ........ 7
2.1 EK Certificates ........ 7
2.2 Platform Certificates ........ 7
2.3 Supported Use Cases ........ 7
2.3.1 Pre-deployment Use Cases ........ 8
2.3.2 Deployment Infrastructure Use Cases ........ 9
2.3.3 Retirement/Redeployment Use Cases ........ 12
2.4 Non-supported Use Cases ........ 12
2.5 Assumptions ........ 12
3 Requirements ........ 13
3.1 General Protocol Requirements ........ 13
3.2 Certificate Policy and Certification Practices ........ 13
3.3 EK CA Requirements ........ 14
3.4 Platform CA Requirements ........ 15
3.5 Non-Requirements ........ 15
4 EK/Platform Enrollment Overview ........ 16
4.1 EK Certificate Enrollment ........ 17
4.1.1 No Existing EK Certificate ........ 17
4.1.2 Existing EK Certificate ........ 18
4.2 Platform Certificate Enrollment ........ 19
4.2.1 Existing EK Certificate ........ 19
4.3 Simultaneous EK and Platform Certificate Enrollment ........ 22
4.4 Virtual TPM Enrollment ........ 22
4.4.1 Simple VMM Proxy Enrollment ........ 23
5 EK/Platform Enrollment Over CMC ........ 24
5.1 CMC Overview ........ 24
5.1.1 Full PKI Request/Response ........ 24
5.1.2 CMC Proof Of Possession (POP) Controls ........ 26
5.1.3 POP Workaround Using Throw-away AIK ........ 27
5.1.4 Putting it together: CMC Enrollment with POP ........ 28
5.2 CMC Authentication Considerations ........ 29
5.3 Implementation Preliminaries ........ 30
5.3.1 Data Security and Encapsulation ........ 31
5.4 The CMC Implementation ........ 32
5.4.1 Creation of the Initial Full PKI Request ........ 32
5.4.2 Creation of the Full PKI Response Without POP ........ 34
5.4.3 Creation of the Initial Full PKI Response With POP ........ 35
5.4.4 Creation of the Full PKI Request Including POP ........ 37
5.4.5 Creation of the Full Final PKI Response with POP ........ 39
6 Security Considerations ........ 41
6.1 Platform Attestation ........ 41
6.2 Certified Keys ........ 41
6.3 Threats to Physical TPM Enrollment ........ 42
6.4 Threats to Virtual TPM Enrollment ........ 43
6.5 Countermeasures ........ 44
7 References ........ 45
7.1 Normative References ........ 45
7.2 Informative References ........ 45

# 1 Introduction

## 1.1 Scope and Audience

This specification describes methods for Endorsement Key (EK) Credential and Platform Credential enrollment. It is frequently assumed that EK and Platform Credentials will be issued during the system's supply chain manufacturing processes (e.g. by the TPM manufacturer and OEM respectively), but in practice, this is often not the case. Since AIK and certified key credential issuance depends on the presence of EK/Platform credentials, it is important to provide guidance to those wishing to issue these credentials at various points in the trusted platform lifecycle. That is the aim of this specification.

Architects, designers, developers, and technologists interested in the development, deployment, and interoperation of trusted platforms will find this document useful and informative. Before reading this document, the reader should review and understand the IWG architecture as described in [1] and [2], and should also read the AIK enrollment specification [4] and the Credential Profiles document [5]

## 1.2 Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

## 1.3 Definitions

This section defines a number of terms and acronyms used in this document. It is important to note that commonly used TCG acronyms (e.g., TCG, TPM, EK, AIK, etc.) are not defined here, but if you read the prerequisite documents referenced in section 1.1, this should not present any difficulties.

| | |
|---|---|
| Device Manufacturer | Used interchangeably with OEM and Platform Manufacturer; see definition for Platform Manufacturer |
| EK CA | The Certification Authority (CA) that issues the Endorsement Key Certificate |
| Enrollment Agent | The EK/Platform certificate enrollment software that interacts with the TPM and the RA/CA when enrolling for EK and/or Platform certificates. |
| OEM | Original Equipment Manufacturer – this term is sometimes used to refer to the platform manufacturer |
| Platform CA | The Certification Authority (CA) that issues the Platform certificate. |
| Platform Manufacturer | This is the entity that assembles the platform elements, including the TPM and TBB |
| privEK | This refers to the private portion of the EK pair. |
| pubEK | This refers to the public portion of the EK pair. |
| Repository services | The EK/Platform CA provides access to its CP, CPS, CRL's, and perhaps OCSP access; these are collectively referred to as Repository Services. |
| TBB | Trusted Building Block |

# 2 Background

## 2.1 EK Certificates

The Endorsement Key (EK) certificate contains the public EK, as well as various assertions regarding the security qualities and provenance of the TPM. Ideally, either the TPM manufacturer or the platform manufacturer provides this credential. However, in many cases, no EK or EK credential is provided for or present in devices when they are delivered to end consumers, so EKs and EK certificates may ultimately be provisioned sometime after manufacturing (e.g., by a deploying IT department).

The EK certificate may be considered to be privacy-sensitive, as in typical deployments, only one EK is created over the lifetime of a TPM, implying that the EK uniquely identifies the TPM (and platform) in which it resides. For this reason, it may be important in some cases to avoid unauthorized disclosure of the EK credential. For additional details on the EK credential see [5].

## 2.2 Platform Certificates

A platform certificate attests that a specific platform contains a unique TPM permanently associated with a static or dynamic root of trust. For comprehensive definitions of these terms, see [5]. The platform certificate is typically issued by the platform manufacturer, and contains a reference to the associated EK certificate, as well as assertions regarding the platform manufacturer, platform model, and platform security properties (among other things). However, like the EK certificate, the platform certificate may also be provisioned sometime after manufacturing.

The platform credential has been specified as an X.509v3 Attribute Certificate (because by definition, it contains no public key), but a lack of widespread attribute certificate support led to the pragmatic compromise of simply making this a standard X.509v3 certificate containing a copy of the EK public key (the Unified Credential [5]).  The latter form is assumed in this document.

## 2.3 Supported Use Cases

At a very high level, this specification aims to provide a general solution to EK/Platform credential enrollment. However, enrollment may occur at various points in the platform lifecycle, and it is important to understand the implications that follow as a consequence. To that end, we review a representative sampling of these various cases below, and from that sampling, define a set of supported use cases.

The TCG Reference Architecture for Interoperability (Part I) [1] defines the trusted platform lifecycle as consisting of a series of stages:

- Manufacturing
- Platform Delivery
- Platform Deployment
- Platform Identity Registration
- Platform Operation
- Platform Recycling/Retirement

While these stages are not all relevant for the purposes of EK/Platform enrollment, this still provides a useful framework for discussion of use cases. Further refining this a bit, the reference architecture document goes on to group these into 3 categories:

- Pre-deployment Infrastructure: consists of entities/functions that support preparation of trusted platforms before they are deployed

    - TPM manufacturers
    - Motherboard suppliers who connect the TPM, TBB and physical presence signals to the board
    - System builders (OEMs/ODMs)
- Deployment Infrastructure: consists of entities/functions that support the use of trusted platforms outside the manufacturing boundary
    - Value-Added Reseller (VAR) and/or System Integrator (SI)
    - Compliance/conformance testing laboratories
    - Organizational Information Technology (IT) department
    - Attestation CA
    - Authentication servers supporting platform authentication
- Retirement/Redeployment Infrastructure: consists of entities/functions that support preparing platforms for retirement (and potentially destruction), as well as for re-purposing. Note that in case of re-purposing, this is really just an extension of the deployment infrastructure category.

Based on the desire to define an EK/Platform enrollment protocol that may be used in any of these stages, we describe some representative use cases from each category, and from these, derive requirements for the enrollment protocol.

### 2.3.1 Pre-deployment Use Cases

There are two general classes of pre-deployment provisioning use cases. In the first, the TPM manufacturer provides the EK and EK certificate, while the OEM provides the platform certificate. In the second, the OEM provides both certificates. These are described in more detail below.

#### 2.3.1.1 EK Certificate Issued by TPM Manufacturer

While a TPM manufacturer might choose to implement a proprietary method for EK enrollment, a standardized approach may provide various advantages. Therefore, while this use case is not a primary motivator for creation of an enrollment protocol, it should be supported nonetheless.

In these use cases, the EK is introduced within the TPM during manufacturing, and the EK certificate is created and issued by the TPM manufacturer. The EK itself may be generated within the TPM, or it may be externally generated and then injected via an API. If it is externally generated, then it seems probable that the EK certificate would simply be generated at the same time as the EK. In that case, no enrollment protocol would be required, although one could be used between the generation agent and the issuing CA, assuming they did not co-reside on the same system.

On the other hand, if the EK is generated within the TPM, the TPM manufacturer may have more reason to implement an enrollment protocol. Since the TPM is not yet installed in a platform, this implies the existence of some sort of external enrollment agent that is integrated into the manufacturing process, able to issue commands to the TPM, and able to carry out enrollment protocol operations on its behalf. For our purposes, the TPM and enrollment agent would be indistinguishable from a TPM and enrollment agent that reside in an already-assembled platform, so associated use cases will be very similar to other pre-deployment use cases with respect to EK enrollment.

#### 2.3.1.2 EK Certificate Issued by OEM

In some cases, the TPM manufacturer chooses not to issue EK certificates, and in such cases, the OEM may choose to issue them instead. There are two cases to account for: in the first, the TPM manufacturer introduces the EK pair into the TPM, but chooses not to issue the EK certificates. In the other, the TPM is delivered to the OEM in an uninitialized state, and the OEM introduces the EK pair into the TPM and provides the associated EK certificate. Both use cases are addressed below.

**R.PD.100**: The enrollment protocol MUST support both serial and batch enrollment modes. Because the EK certificate may be issued prior to platform certificate enrollment, the enrollment request MUST provide a method for referring to an existing EK certificate. Likewise, because the EK certificate may not have been issued prior to platform certificate enrollment, a batch enrollment mode (allowing simultaneous enrollment for EK and platform certificates) is required.

When the TPM manufacturer generates the EK, it is conceivable that in some cases the OEM would be provided with a list of EKs against which subsequent enrollment requests could be validated. Such functionality is useful both for security and quality control purposes. However, it also implies that an enrollment request could be rejected because the associated EK is “wrong”, either because it is unexpected, or because it has previously been enrolled. A well-designed enrollment protocol will provide clear notification of the associated error condition.

**R.PD.110**: The enrollment protocol MUST provide an unambiguous error indication when an EK enrollment request is rejected because the EK already exists.

**R.PD.120**: The enrollment protocol MUST provide an unambiguous error indication when an EK enrollment request is rejected because the EK is unexpected/unauthorized.

#### 2.3.1.3 Platform Certificate Issued by OEM

The platform certificate, because it contains a reference to the EK certificate, cannot be created prior to creation of the EK certificate. Furthermore, because the platform certificate contains assertions about the platform in which the TPM resides, the OEM is the most logical source for this certificate. The term “OEM”, when used here, may apply to the entity that creates the TBB, or to the entity that integrates the TBB into the final platform. For our purposes, we will treat these cases as equivalent.

In some cases, the OEM will issue both the EK and platform certificates in one enrollment operation. In others, the EK certificate has been issued in advance (e.g., by the TPM manufacturer). In the former case, the enrollment request contains a Certificate Signing Request (CSR) for the EK, whereas in the latter case it contains the EK certificate rather than the CSR. The enrollment protocol MUST support either approach (R.PD.100).

### 2.3.2 Deployment Infrastructure Use Cases

As noted above, deployment infrastructure encompasses entities/functions that support the use of trusted platforms outside the manufacturing boundary. In terms of the platform lifecycle, this includes platform delivery, deployment, identity registration, and operation. This is a very broad area that includes the bulk of the use cases addressed by this specification.

In all of these use cases, is important to note that there are 3 possibilities with respect to the EK:

1. The EK was introduced into the TPM sometime during the pre-deployment phase, and this EK will be utilized as-is.

2. The EK was introduced into the TPM sometime during the pre-deployment phase, but it will be replaced (i.e. deleted and re-generated). Note that a limited subset of TPMs support this functionality.
3. No EK exists; it must be generated prior to creation of the EK/platform certificates.

In addition, in cases where the EK was generated during the pre-deployment phase, platform and/or EK certificates may have been issued by someone in the pre-deployment supply chain; however, there may be valid reasons for not utilizing these, even though the EK itself will not be replaced. Each of the use cases below must take these things into account.

#### 2.3.2.1 Issuance of EK/Platform Certificates by Service Provider

A service provider may prepare a platform for use by subscribers, where an integral element of that preparation includes TPM initialization and provisioning. In such cases, the service provider may support EK/Platform certificate enrollment as part of the provisioning process. In some cases, the EK Certificate may already be present, having been included by the TPM manufacturer; in these cases, the provider may provision only the Platform certificate. Alternatively, the service provider may choose to replace the EK and/or the EK/Platform certificates issued earlier in the supply chain with its own.

In cases where certificates are being replaced, the service provider may wish to rely on assertions present in the previously supplied certificates. In such cases, the simplest way to communicate the associated assertions, along with the information needed to verify their source, is to include the entire previously issued certificates in the enrollment request.

**R.DI.100**: the protocol MUST support inclusion of previously issued certificates in the enrollment request.

#### 2.3.2.2 Issuance of EK/Platform Certificates by a Conformance Evaluator

In some cases it is desirable to have various aspects of a platform evaluated for conformance with some criteria, and the conformance evaluator may issue EK and/or Platform certificates containing assertions that are based on this evaluation. This use case is very similar to the service provider use case described above, except that the conformance evaluator will typically strive for more extensive validation prior to issuing credentials.

#### 2.3.2.3 Issuance of EK/Platform Certificates by a Value-Added Reseller

A value-added reseller may choose to activate the TPM and provision associated certificates as a precursor to some related feature they are providing, such as a Full Disk Encryption. This use case is similar to the service provider use case, with the primary difference being in who “owns” the TPM. In the service provider use case, it is the provider; the TPM is wholly used and managed by the service provider. In the VAR use case, the TPM may be used for other features as well, and TPM ownership ultimately resides with the end user (or the end user’s agent).

#### 2.3.2.4 Issuance of EK/Platform Certificates by an Organizational Entity

It is frequently the case that new systems arrive in the organizational IT department with TPMs that have not been activated or provisioned. Having the ability to initialize the TPM and provision EKs and EK and Platform certificates to such systems is a necessary first step toward utilizing the TPM for various security-enhancing functions. In some cases, provisioning will occur before the system is delivered to the end user. However, given that there are millions of TPM-containing systems currently deployed, it would also be very useful to have a method for “field” provisioning of EK/Platform certificates.

In cases where provisioning occurs prior to delivery to the end user, one may envision a "compact" process in which the CA signing keys are stored in, e.g., a USB-based cryptographic device that is connected to the system and utilized by a transient initialization/enrollment application that runs directly on the provisioned system. In such cases, an enrollment protocol is clearly not required. However, there may be key management, scaling, operational, and/or security requirements that motivate use of an external RA/CA infrastructure instead. Indeed, in the case of field provisioning, this would almost certainly be required. In such cases, it may be desirable to "validate" the enrollment transaction in various ways. This suggests that a flexible enrollment protocol, one that accommodates generalized validation plug-in functionality, would be useful. However, we do not impose any related requirements in this specification.

#### 2.3.2.5 Issuance of EK/Platform Certificates for Virtual Machines

The TCG Virtualization Work Group has defined methods for implementing virtual TPMs ("vTPMs")[17]. The Virtual Machine Manager (VMM) may create a vTPM when a Virtual Machine (VM) is first instantiated. While it is certainly true that first instantiation is a logical time to provision EK and platform certificates, it may be that the particular VM being created does not require them, e.g. because it will not make use of the vTPM. There may alternatively be a desire to place provisioning under the control of the VM owner. In these latter cases, enrollment at first instantiation is neither desired nor required.

In cases where enrollment *does* occur at instantiation, one may envision the VMM acting as the issuing CA, perhaps utilizing a TPM-resident signing key that resides in the physical TPM associated with the platform. Such a key could be migrated to (shared with) other VMMs, essentially creating a distributed CA. In such cases, an enrollment protocol is clearly not needed: the VMM simply issues certificates as part of the VM creation process.

Alternatively, the CA may be external to the VMM, in which case some sort of enrollment protocol is desirable. For these cases, we may envision at least two general enrollment scenarios:

1. The VMM creates the vTPM, and acts as the enrollment agent on behalf of the VM. In this case, the VMM presumably has access to a physical TPM. There are various ways in which this physical TPM could be used to facilitate attestation of the enrollment process. For example, suppose the VMM has a TPM-resident key with which it can sign enrollment requests, or, that it has an AIK that can be utilized for attestation prior to issuance of the vTPM credentials. Reliance on such capabilities has definite implications for what would be carried in an enrollment request/response exchange.
2. The VMM creates the vTPM and instantiates the VM. The VM "install" image includes an enrollment agent that runs at first boot, and it initializes the vTPM, generates the EK, and executes the enrollment protocol. In such cases, the enrollment agent may or may not take into account the fact that it is in a VM. In either case, the RA/CA may require some sort of VMM attestation (or other interaction) in order to complete the enrollment. In cases where the VM is unaware of the VMM, this would have to be managed out of band, but in cases where the VM (or at least, the VM install image) *is* aware of the VMM, some sort of VMM attestation could be included in the enrollment exchange.

In this specification, we address only the first of these two scenarios, the one in which the VMM enrolls on behalf of the VM. The second scenario is addressable through a combination of out of band attestation of the VMM and use of a more generalized enrollment protocol (similar to one used for a non-virtualized host), so we do not address that separately here.

### 2.3.3 Retirement/Redeployment Use Cases

#### 2.3.3.1 Re-Issuance of EK/Platform Certificates

In terms of redeployment, there are two general cases we need to address. In the first, the TPM supports deletion of the EK. This is an optional feature that is not supported by many currently deployed TPMs. In this case, the old EK is purged, a new EK is generated, and new certificates are issued. In the second case, the EK is re-used, old certificates are discarded, and new certificates are issued.

The first case is very similar to a “fresh” enrollment scenario, except that it offers the possibility of using the original EK in the enrollment exchange so that security properties could be copied from the original EK certificate. However, since the original EK must be deleted from the TPM prior to creation of the new one, this leads to protocol complexities that are difficult to justify. For that reason, such functionality is not supported by this specification. The second case, on the other hand, may allow for copying the security properties into the new certificate, provided that trust in the original EK certificate can be established. This functionality should be supported.

## 2.4 Non-supported Use Cases

The following use cases are not covered by this version of the EK/Platform certificate enrollment protocol:

- Protocols other than CMC are not supported
- Platform attestation during the enrollment process (e.g. using TNC protocols) is not covered by this specification.

## 2.5 Assumptions

- The credentials referenced by this specification conform to those described in [5]
- The enrollment agent is trusted to make truthful assertions with respect to the EK being presented for certification, and with respect to the TPM within which the EK resides. It is assumed that enrollment operations are carried out under conditions that facilitate and justify such trust. See security considerations section for additional discussion.

# 3 Requirements

## 3.1 General Protocol Requirements

In reviewing the various use cases above, we can make a number of observations:

- In some cases, enrollment occurs in a strictly controlled network environment. In such cases, confidentiality is not a concern, nor is protocol endpoint authentication. However, in other cases enrollment may occur over a potentially hostile network. In those cases, confidentiality and authentication are both important. Therefore, the enrollment protocol must support these.
  - **R.GPR.100**: The protocol MUST provide native support for confidentiality
  - **R.GPR.110**: The protocol MUST provide native support for strong, mutual authentication
- In some cases, EK and platform certificates are issued independently, perhaps at different stages in the trusted platform lifecycle, while in others, they are issued simultaneously. This implies that the enrollment protocol should support both cases. While it could be argued that simply supporting independent EK and platform certificate enrollment implies support for both (in series), it is simpler and more efficient to provide for an atomic enrollment operation in cases where both certificates are required, so we should strive to achieve this (covered by R.PD.100).
- In some cases, it may be desirable to utilize an existing EK/platform certificate to “bootstrap” the enrollment process. For example, a VMM enrolling on behalf of a newly created vTPM might include its own EK/platform certificates in the process. Other examples include a platform that is renewing an expiring certificate, or a service provider or VAR who is providing certificates for its own closed domain, but who is willing to copy information from existing certificates.
  - **R.GPR.120**: The enrollment protocol MUST support replacement of an existing EK certificate that is passed in an enrollment request.
- In some cases, requiring proof of (private key) possession prior to issuing a certificate may be advisable from a security perspective. The protocol must provide optional support for this.
  - **R.GPR.130**: The protocol MUST support Proof of Possession (POP) for the EK.

In general, we are motivated to re-use existing technologies where possible, due to the various benefits this yields. In considering choices for EK and Platform certificate enrollment, the fact that CMC has already been standardized by TCG as an enrollment protocol for AIKs creates a strong incentive to adopt CMC for this effort as well. In addition, CMC easily meets the various specific and general protocol requirements outlined above.

## 3.2 Certificate Policy and Certification Practices

If EKs, EK certificates, and Platform certificates are to be used in an interoperable manner, then the EK and Platform CAs must provide formal statements upon which potential enrollment domain participants can base expectations and trust. Typically, this is accomplished through publication of Certificate Policy (CP), and/or a Certification Practices Statements (CPS). While current X.509 standards don’t explicitly mandate such documents, this particular specification requires some sort of structured statement for this purpose. In the interest of flexibility, the exact form of this statement is not specified here. Nonetheless, for simplicity, we refer to this documentation below as “the CP/CPS”.

Ultimately the CA will provide a policy statement that, paraphrasing [15], describes the applicability of issued certificates to a particular community or class of applications. In addition, the CA should describe its issuance policies and operational practices. It is on the basis of such statements that a relying party and enrollee invest trust in the CA.

**R.CPCPS.100**: The CA SHOULD provide formal statements describing enrollment policies, issuance policies, and operational practices. This statement MAY take the form of a CP and/or CPS, or it may take some other form. Regardless, in statements below this is referred to as the CP/CPS.

**R.CPCPS.110**: The Authority Information Access (AIA) extension in the EK/Platform certificate(s) SHOULD facilitate retrieval of the CP/CPS.

In order to simplify the enrollment protocol, this specification calls for the EK/Platform CA to choose which cryptographic algorithms it will support. In order for implementers to have some indication of which algorithms to implement, these must be specified somewhere. This will be included in the CP/CPS.

**R.CPCPS.120**: The CP/CPS SHOULD indicate which cryptographic algorithms the EK/Platform CA supports for enrollment.

## 3.3 EK CA Requirements

Requirements for the EK CA will vary depending on several factors:

- At what point in the TP lifecycle is the EK generated?
- Will the EK be used for more than a single application, and if so, are all of these applications managed by one provider, or are there potentially multiple, independent providers?
- Is the Platform CA under the control of the same entity who controls the EK CA? Another way to frame this would be to ask: must the EK certificate be useful for inter-domain applications?

These points hint at several fundamental issues relating to EK provisioning whenever multiple TPM applications might be installed on a single platform, either together or in succession. Put simply, if later providers cannot trust assertions about EK generation, and/or the circumstances do not meet their requirements, it may not be possible for these applications to co-exist. Furthermore, if an existing EK cannot be deleted and re-generated (either because the TPM does not support this, or because to do so would “break” existing applications), this again may constrain or preclude the installation of later applications.

The point in the TP lifecycle where provisioning occurs also has a strong influence on EK CA requirements. The earlier in the lifecycle that EK generation and certificate provisioning occurs, the less that is known about potential use cases. For example, when a TPM manufacturer introduces EKs into TPMs and provides the corresponding EK certificates, there are numerous potential uses, and the TPM manufacturer should be motivated to provision in a manner that is applicable to the broadest number of potential use cases.

Given the many ways in which EKs and EK certificates might be provisioned and used, and given our limited experience with this so far, it seems prudent to provide guidelines here, but to avoid specifying rigid requirements that might turn out to be overly restrictive. To that end, and based on the discussion above, we provide a small number of general guidelines below.

- **R.EKCA.100**: the EK CA SHOULD provide reliable, accessible repository services in order to facilitate access by and smooth operation of independently controlled Platform and/or Attestation CAs.
- **R.EKCA.110**: the CP/CPS SHOULD describe the procedures relating to external EK generation/introduction. In particular, if the EK is generated external to the TPM, then the CP/CPS SHOULD describe
  - How the EK pair is generated
  - How the privEK is protected against unauthorized disclosure prior to introduction into the TPM
  - If and how copies of the privEK are destroyed following introduction into the TPM
  - How copies of the privEK are protected, if they are not destroyed, and procedures for retrieval/distribution of retained privEK values
- **R.EKCA.120**: the CP/CPS SHOULD describe the privacy policy relating to EKs, including statements regarding retention of and release of information relating to EKs, enrollment, etc.

## 3.4 Platform CA Requirements

Like the EK CA, requirements for Platform CAs will vary depending on a number of factors. In cases where a single provider controls the EK, Platform, and Attestation CAs, the requirements will be entirely dictated by the use case. However, like the cases for the EK CA, it becomes more interesting when there are “upstream” providers (e.g. an ACA) that fall within an independent domain of control with respect to EK/Platform certificates. In such cases, the upstream provider will want to fully understand the basis for the assertions contained in the Platform certificate, rather than simply accepting them at face value.

While the idea of a scalable, generalized solution is appealing, the reality is that this is probably not possible. When a Platform certificate is issued in pre-deployment scenarios, we might assume that the issuer has a strong basis for making security assertions about the platform, but in situations requiring any significant level of assurance, some sort of policy decision must be made, one that likely requires human intervention, and the upstream RAs/CAs must be configured accordingly.

Given this, the requirements for the Platform CA will fall mostly around the mechanics of certificate validation during the AIK enrollment process:

- **R.PCA.100**: the Platform CA SHOULD provide reliable, accessible repository services in order to facilitate access by and smooth operation of independently controlled Attestation CAs.

## 3.5 Non-Requirements

- Definition of requirements and procedures for TPM/Platform verification is explicitly out of scope

# 4 EK/Platform Enrollment Overview

Based on the requirements derived from the use cases described above, we expect the contents of the enrollment transactions to vary somewhat depending on specifics of the use case. Before we get into the complexities of CMC, it is useful to spend some time here understanding what is required in each part of the exchange, and why. Once this is done, we can dig into how to layer the required functionality into CMC.

In the following enrollment scenario outline, we distinguish between virtualized vs. non-virtualized TPMs; please note that virtualized TPM use cases are not supported for all enrollment variants. The following scenarios are supported for non-virtualized TPMs:

- Enroll for EK certificate (4.1)
    - With no existing EK certificate (4.1.1)
    - With existing EK certificate (4.1.2)
- Enroll for Platform Certificate (4.2)
    - With existing EK certificate (4.2.1)
        - With existing Platform certificate (4.2.1.1)
- Simultaneously enroll for EK/Platform Certificates (4.3)

Note that in case of simultaneous enrollment for EK and Platform certificates, there are several potential variations, based on whether or not there are already existing EK and Platform certificates. Dealing with these cases here would add significant complexity to the protocol. Given that we don't know how useful this capability will be, and that the associated use cases can also be addressed by enrolling serially for EK and then Platform certificates, we do not support the more complex simultaneous variants in this specification. If sufficient demand arises, the protocol can be extended to support these at a later date.

In addition to the non-virtualized TPM scenarios described above, we also discuss several vTPM enrollment scenarios below (4.4). Since this specification only describes VMM-based vTPM enrollment, we only discuss two variants:

- When the vTPM is created, the VMM simultaneously enrolls for both certificates (4.4.1, 4.4.2)
- When the vTPM is migrated
    - New host VMM may replace the Platform certificate alone (4.4.3)

Note that if the VMM replaces both certificates upon migration, this is equivalent to a fresh enrollment, so we do not discuss this separately below. Also note that the VMM will never enroll for or replace just the EK certificate alone. It will either be enrolling for both (at creation), replacing both (at migration), or replacing the Platform Certificate (also at migration). Therefore, enrollment for EK certificates alone is not supported for vTPMs. Each of the scenarios outlined above is described below in sections corresponding to the value in parentheses following the outline descriptions above.

## 4.1 EK Certificate Enrollment

This scenario describes enrolling for an EK certificate. There are two sub-cases:

- no existing EK certificate is used in the enrollment request
- there is an existing EK certificate which is included in the enrollment request

We would expect that typically, there is no existing EK certificate. However, in cases where there is one, there may be a desire to rely on the assertions within that certificate. For that reason, we support inclusion of that certificate within the enrollment request.

### 4.1.1 No Existing EK Certificate

There are two EK enrollment use cases possible when there is no existing EK certificate: with and without EK Proof of Possession (PoP). With PoP, the enrollment exchange is as follows:

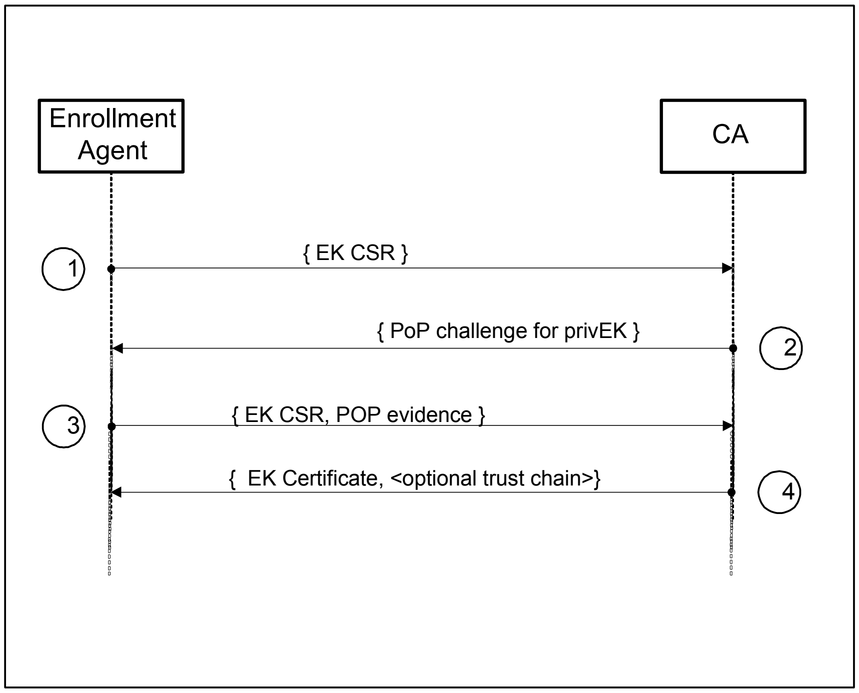

Enrollment Agent
CA
1
{ EK CSR }
{ PoP challenge for privEK }
2
3
{ EK CSR, POP evidence }
{ EK Certificate, <optional trust chain>}
4

**Figure 1 - Basic EK enrollment with PoP**

In the first message, the enrollment agent provides the CSR, which contains the pubEK and TPM/TBB assertions. The RA wants to verify that the enrollment agent can utilize the associated privEK (proof of EK possession, or PoP), so it replies with a PoP challenge (2). The enrollment agent constructs the PoP evidence, and resubmits the request (3). The CA then returns the issued EK certificate, and optionally includes the associated trust chain (4).

One important design decision surfaces from this overview: who encodes the TPM and TBB security assertions for inclusion in the certificate? Is it the enrollment agent, or is it the CA? If we assume that the enrollment agent will construct a Certificate Signing Request (CSR), then the protocol will be arguably simpler if the agent encodes the various assertions in their final form and includes them in the CSR. This is the approach taken below.

In case PoP is not required, the enrollment exchange will proceed as follows:

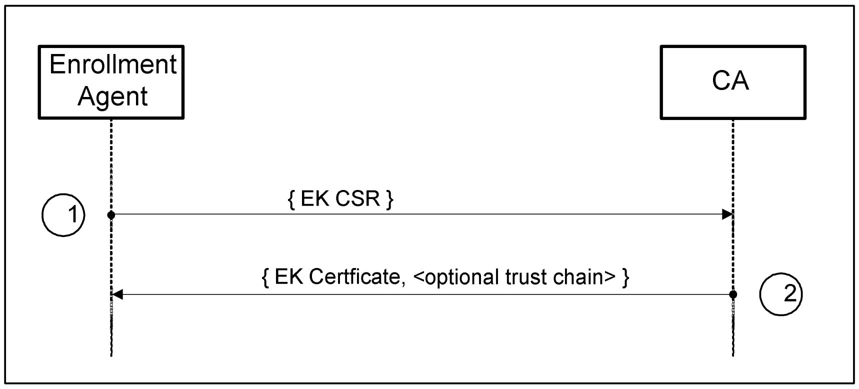


**Figure 2 - Basic EK Enrollment without PoP**

### 4.1.2 Existing EK Certificate

The primary difference between 4.1.2 and 4.1.1 is that in the latter case, an existing EK certificate is included in the enrollment request. There are two reasons why this might be useful. The first is that the RA/CA can compare the EK in the request with the one in the certificate. That is, in deployment infrastructure use cases, this is one of the few ways to gain additional assurance that the key being enrolled truly is a TPM-resident EK.

The second reason the existing certificate might be useful is that it provides a way for the CA/RA to compare the TPM and TBB security assertions in the CSR with the ones in the existing certificate. In many cases this provides no additional value, but it is convenient, and in some cases it may provide some additional assurance.

It's important to note that in these cases, the RA/CA must have the ability to validate the existing EK certificate, and this implies that it has access to the associated trust chain. In this specification, we assume that trust chain is somehow obtained out of band. The resultant exchange looks like this:

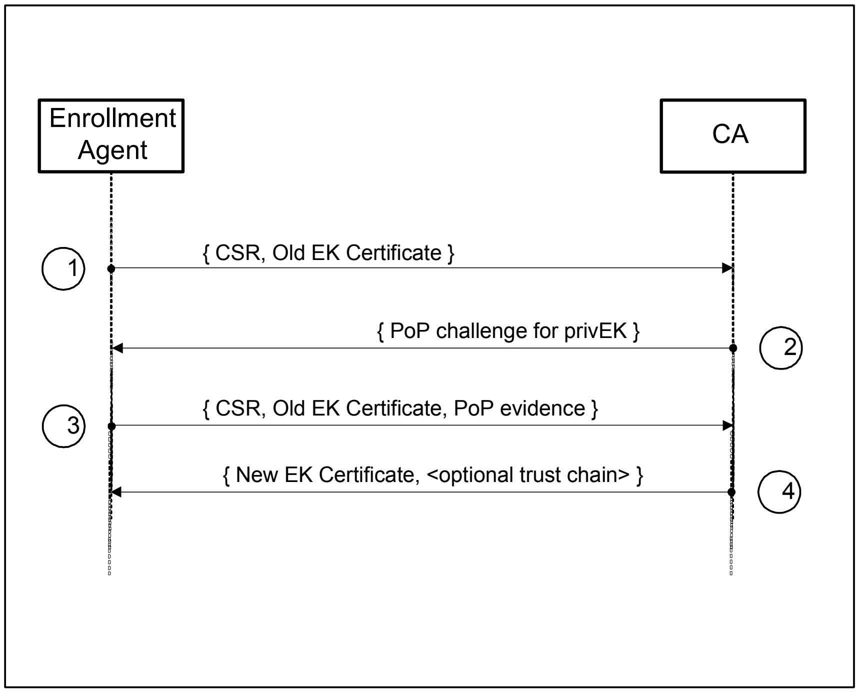

Enrollment Agent
CA
1
{ CSR, Old EK Certificate }
{ PoP challenge for privEK }
2
3
{ CSR, Old EK Certificate, PoP evidence }
{ New EK Certificate, <optional trust chain> }
4

**Figure 3 - EK enrollment with Existing Certificate and PoP**

In the case where PoP is not desired, the simplified exchange looks like this:

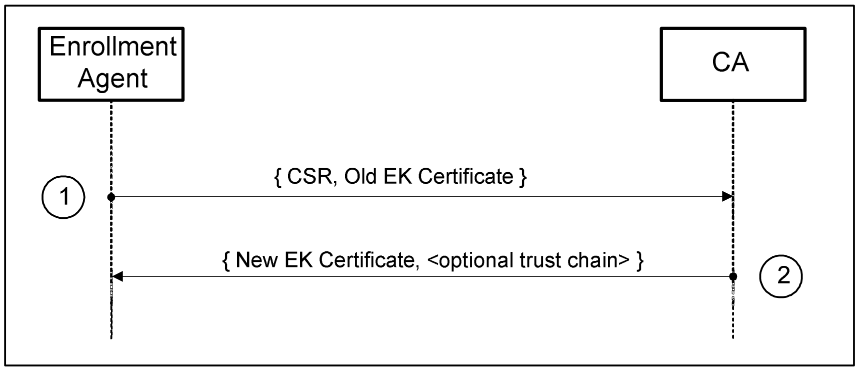

Enrollment Agent
CA
1
{ CSR, Old EK Certificate }
{ New EK Certificate, <optional trust chain> }
2

**Figure 4 - EK Enrollment with existing EK Certificate**

## 4.2 Platform Certificate Enrollment

Note that platform certificate enrollment cannot proceed unless there is an existing EK certificate. Therefore, all Platform Certificate enrollment exchanges include an existing EK certificate.

### 4.2.1 Existing EK Certificate

For Platform certificate enrollment, the exchange is as follows:

In some cases, we may wish to verify EK PoP prior to issuing the Platform certificate. In such cases, the exchange is almost identical to the EK enrollment exchange:

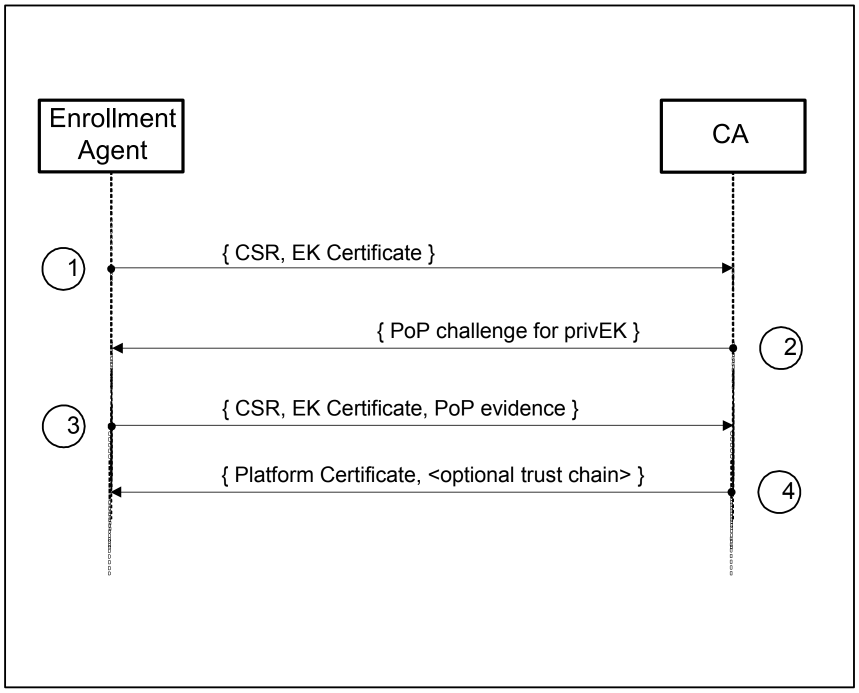

Enrollment Agent
CA
1
{ CSR, EK Certificate }
{ PoP challenge for privEK }
2
3
{ CSR, EK Certificate, PoP evidence }
{ Platform Certificate, <optional trust chain> }
4

**Figure 5 – Basic Platform Certificate Enrollment with EK PoP**

If no EK PoP is required, the platform certificate enrollment exchange is significantly simplified:

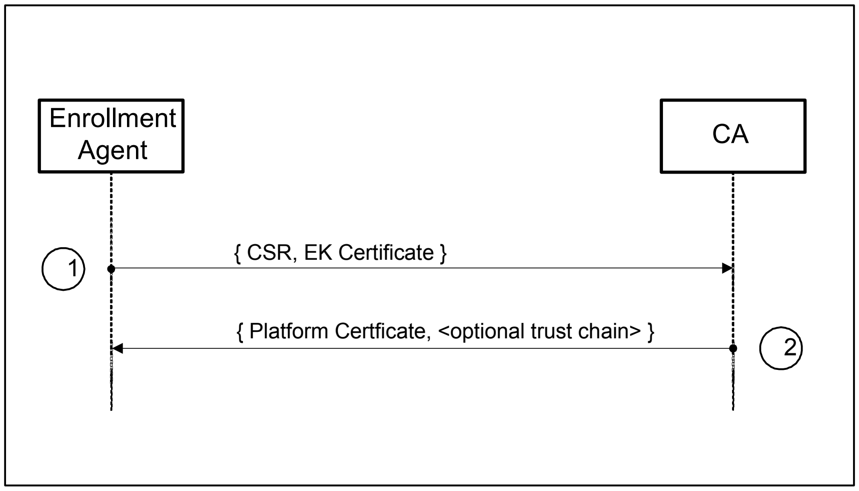


**Figure 6 - Basic Platform Certificate Enrollment**

As with the EK enrollment exchange, note that the enrollment agent encodes and includes the required platform assertions in the platform certificate CSR.

#### 4.2.1.1 Existing Platform Certificate

In case we want to include an existing platform certificate in the enrollment request (e.g., so that the RA can validate the assertions included in the CSR), then the basic enrollment request is identical to that of 4.2.1, except that we now include the existing platform certificate. If EK PoP is desired, the exchange will look like this:

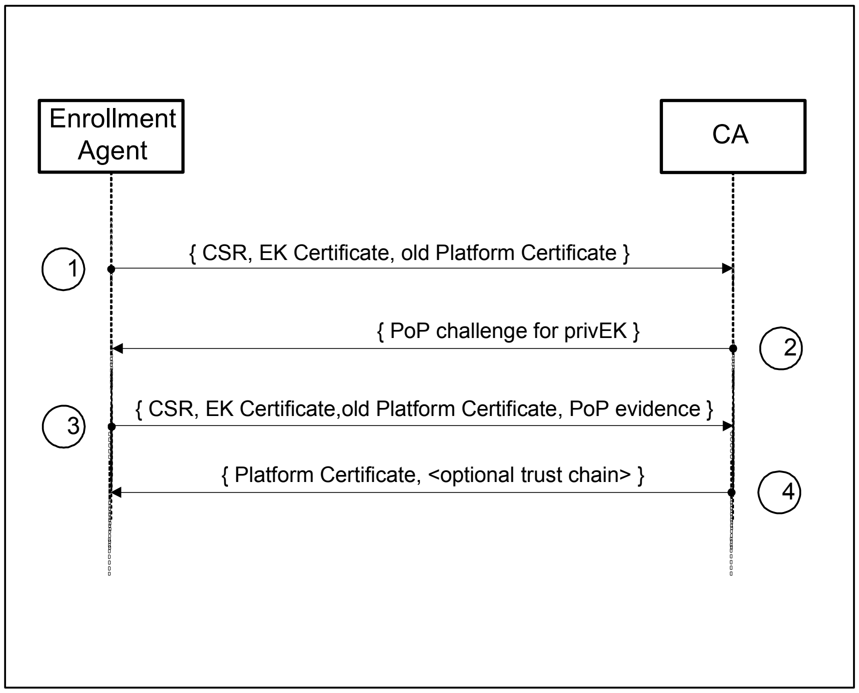

Enrollment Agent
CA
1
{ CSR, EK Certificate, old Platform Certificate }
{ PoP challenge for privEK }
2
3
{ CSR, EK Certificate,old Platform Certificate, PoP evidence }
{ Platform Certificate, <optional trust chain> }
4

**Figure 7 - Platform Certificate Enrollment with Existing Certificate and PoP**

Note that in this case, like that of the EK certificate, we assume that the RA has acquired the associated trust chain through out of band means. If EK PoP is not desired, the exchange will instead look like this:

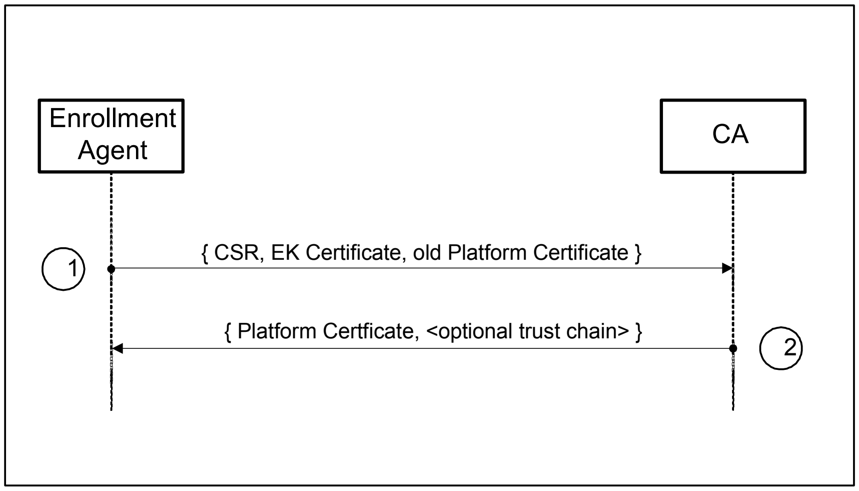

Enrollment Agent
CA
1
{ CSR, EK Certificate, old Platform Certificate }
{ Platform Certficate, <optional trust chain> }
2

**Figure 8 – Platform certificate enrollment with existing certificate**

## 4.3 Simultaneous EK and Platform Certificate Enrollment

The current specification supports only the simplest enrollment variant for simultaneous enrollment: we assume that there are no existing EK/Platform certificates. Assuming EK PoP is desired, the enrollment exchange proceeds as follows:

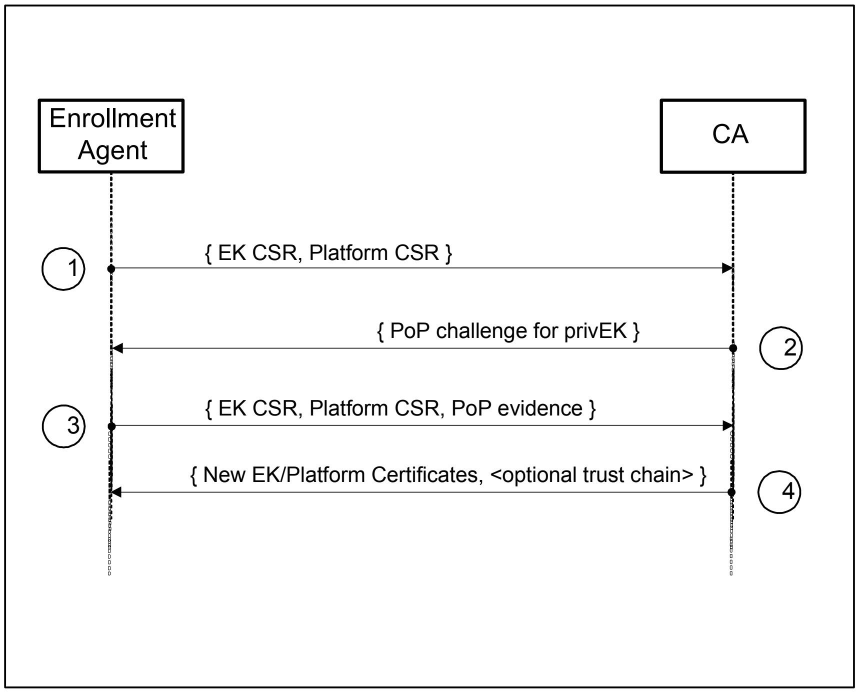

Enrollment Agent
CA
1
{ EK CSR, Platform CSR }
{ PoP challenge for privEK }
2
3
{ EK CSR, Platform CSR, PoP evidence }
{ New EK/Platform Certificates, <optional trust chain> }
4

**Figure 9 - Combined EK/Platform enrollment (basic exchange) with PoP**

If no PoP is desired, the simplified exchange looks like this:

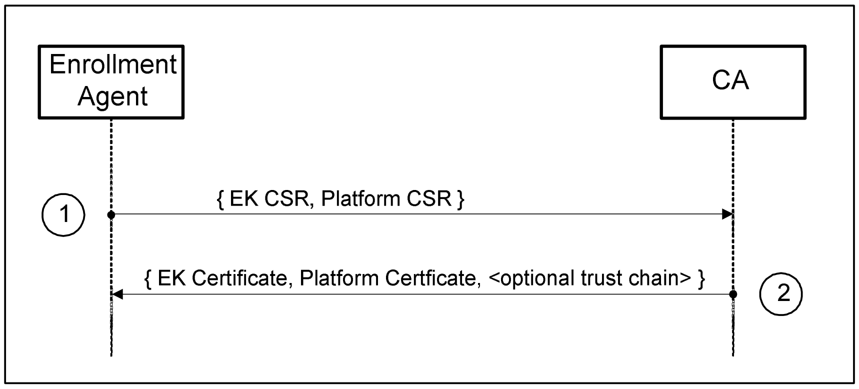

Enrollment Agent
CA
1
{ EK CSR, Platform CSR }
{ EK Certificate, Platform Certficate, <optional trust chain> }
2

**Figure 10 - Combined EK/Platform enrollment (basic exchange)**

## 4.4 Virtual TPM Enrollment

In the simplest case, enrollment of a virtual TPM (vTPM) is indistinguishable from that of a physical TPM. In such cases, a VM-resident enrollment agent may accomplish vTPM enrollment in the same manner as in a non-virtualized case, and the agent may or may not be aware of the fact that it resides in a VM. However, if the VMM has access to a previously activated and provisioned physical TPM, we can make use of this when the VMM acts as an enrollment agent for the vTPM.

There are several advantages to this VMM-based approach. For example, if the physical TPM associated with the VMM has been provisioned with an AIK, the VMM can provide robust attestation as part of the enrollment process. In addition, while we may have to utilize software-based authentication keys for client-side authentication in the case of simple VM-based EK/Platform enrollment, the VMM may utilize a TPM-resident “Certified Key” [14] for client-side authentication. This potentially provides for a significantly higher level of assurance than may be obtained in other post-manufacturing enrollment scenarios.

This specification currently describes only one approach to VMM proxy enrollment, where the VMM enrolls on behalf of the VM, and any attestation occurs either implicitly, or out of band. If, once the protocol is deployed, there is sufficient demand for explicit in-band attestation, then the protocol can be extended to accommodated this.

### 4.4.1 Simple VMM Proxy Enrollment

In this use case, the VMM enrolls on behalf of the vTPM. This is very similar to the (non-virtualized) simultaneous EK and Platform certificate enrollment described above, except that the enrollment request must also contain the certificate matching the key that the VMM uses to sign the enrollment request. The exchange is illustrated in the following figure.

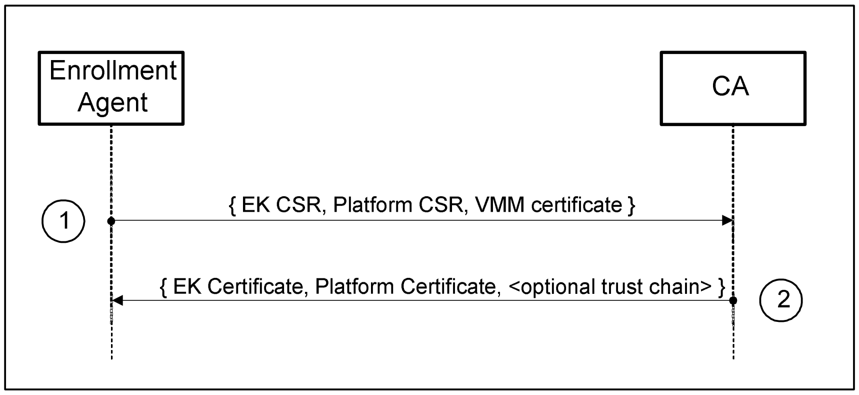

Enrollment Agent
CA
1
{ EK CSR, Platform CSR, VMM certificate }
{ EK Certificate, Platform Certificate, <optional trust chain> }
2

**Figure 11 - Simple VMM Proxy Enrollment**

Note that if the key the VMM signs the enrollment request with is a “Certified Key” [14] that is sealed to the desired platform state, this provides a form of implicit attestation. If the attested VMM code requires that the key be reloaded for each vTPM enrollment request (thereby ensuring that the implicit attestation is “fresh”), this is functionally equivalent to an explicit attestation exchange. However, since implementation of Certified Keys may impose significant additional overhead, explicit attestation may be preferable in some cases.

# 5 EK/Platform Enrollment Over CMC

In this section we give a brief overview of CMC [6][7] followed by a description of how EK and Platform certificate enrollment above may be implemented over CMC. Note that this section is very terse in its description of CMC, assuming that the reader either has the necessary background, or will read the referenced documents for additional information.

## 5.1 CMC Overview

Certificate Management Messages Over CMS (CMC) is a comprehensive, standardized certificate management protocol. While every effort is made to align this proposal with the protocol requirements described in [6][7], in some cases we impose restrictions or deviations, which might be considered non-compliant with the IETF standard. We do this only when strictly necessary in order to (a) maintain compatibility with existing TCG specifications, and/or (b) support a higher level of assurance consistent with TCG applications.

CMC supports 3 high-level enrollment exchange types, each composed of PKI Requests and PKI Responses: exchanges using the Simple PKI Request/Response, exchanges using the Full PKI Request and a Simple PKI Response, and exchanges using the Full PKI Request/Response. CMC also supports numerous other functions, but we leave them aside for now. PKI Requests used for enrollment are in part formed using either the PKCS #10 [9] or CRMF structure. Following is a brief summary of the supported request types:

- Simple PKI Request: a bare PKCS #10 request with no CMS elements
- Full PKI Request: one or more PKCS #10, CRMF, or Other Request Message structures wrapped in a CMS encapsulation as part of a PKIData content-type element.

PKI responses are based on the CMS [8] SignedData element. Following is a brief summary of the supported response types:

- Simple PKI Response: SignedData containing only certificates (e.g. the requested certificate, and the associated trust anchor chain)
- Full PKI Response: a PKIResponse wrapped in a SignedData.

The Simple PKI Request is suitable for straightforward enrollment scenarios involving signed PKCS #10 structures, and in general, is used in a single round-trip exchange. The Full PKI Request, on the other hand, is potentially very rich and complex. Note that in general, use of the Simple PKI Request is not suitable for EK enrollment because the EK is an encryption-only key, and cannot be used to produce the required signature on the request. Therefore, the Simple PKI Request is not supported by this specification.

### 5.1.1 Full PKI Request/Response

Following is a high-level overview of the content of the Full PKI Request and Response as defined for CMC. This is for illustrative purposes, and is simplified accordingly. For complete details, see [7].

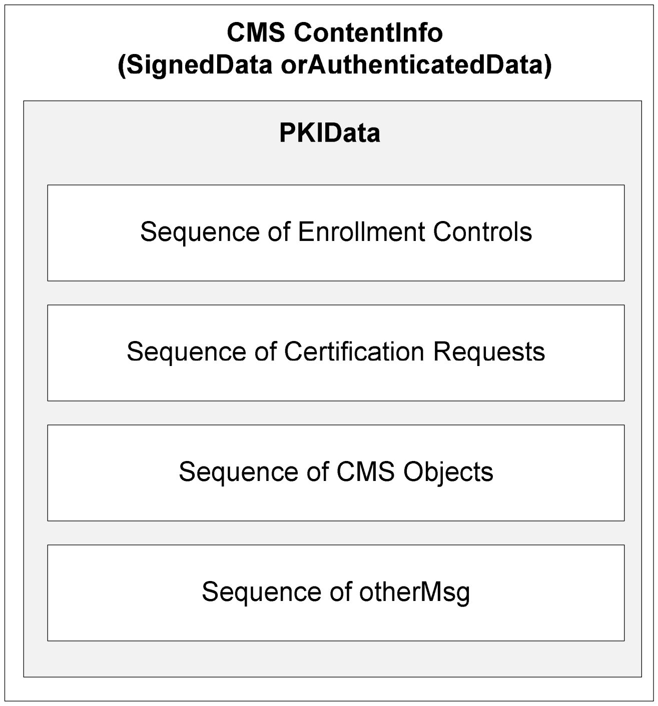

CMS ContentInfo
(SignedData orAuthenticatedData)
PKIData
Sequence of Enrollment Controls
Sequence of Certification Requests
Sequence of CMS Objects
Sequence of otherMsg

**Figure 12 - Full PKI Request**

Note that the request contains the following (in the PKIData structure):

- Enrollment control sequence
    - A sequence of zero or more enrollment controls as defined in CMC
- Certification request sequence
    - A sequence of zero or more certification requests based on PKCS #10, CRMF, or Other Request formats.
- CMS object sequence
    - A sequence of zero or more CMS message objects. The four content types used are AuthenticatedData, Data, EnvelopedData, and SignedData (defined in [6]).
- Other message sequence
    - A sequence of zero or more arbitrary data objects which are referred to by one or more controls (allows controls to use large amounts of data without having to embed the data directly in the controls).

The following illustrates the full PKI response:

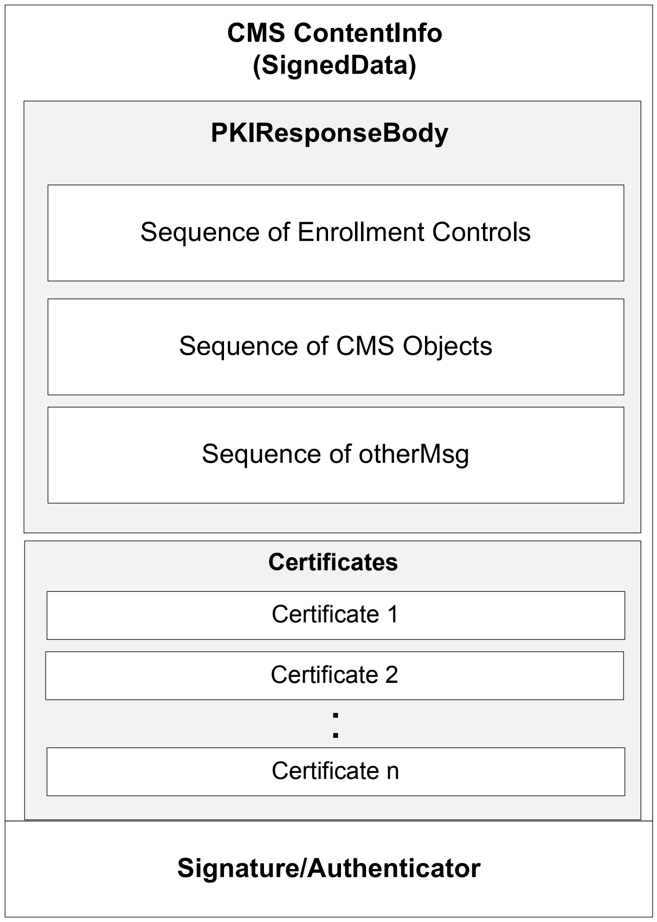


**Figure 13 - Full PKI Response**

The response contains the following in the PKI Response Body:

- Enrollment control sequence
    - A sequence of zero or more enrollment controls as defined in CMC
- CMS object sequence
    - A sequence of zero or more CMS message objects. The four content types used are AuthenticatedData, Data, EnvelopedData, and SignedData (defined in [6]).
- Other message sequence
    - A sequence of zero or more arbitrary data objects which are referred to by one or more controls (allows controls to use large amounts of data without having to embed the data directly in the controls).

Note that the issued certificates are included in the Certificates portion of the SignedData.

### 5.1.2 CMC Proof Of Possession (POP) Controls

In some cases, the Endorsement CA will require proof of possession of the EK private key, but since the EK cannot be used for signing, an alternative method is required. CMC supports POP for encryption-only keys either through proving knowledge of a shared secret, or via use of the Encrypted and Decrypted POP controls. Since the objective is explicit proof of possession for the EK, we must rely on the latter.

For encryption-only keys, there are 4 distinct steps required for explicit POP:

1. Client tells server about public component of encryption key pair, typically as part of a certification request transaction
2. Server sends client a POP challenge, encrypted with the public encryption key
3. Client decrypts POP challenge with corresponding private key and sends proof to server
4. Server validates proof and continues processing certification request

CMC defines two relevant POP-related controls: one for the encrypted “challenge”, sent from the server to the client, and one for the “proof”, sent from the client to the server.

The Encrypted POP control is used to send the encrypted challenge from the server to the client as part of the PKIResponse. Note that it is assumed that the message sent in Step 1 above is a Full PKI Request and that the response in step 2 is a Full PKI Response including a CMCFailInfo specifying that a POP is explicitly required, and providing the POP challenge in the encryptedPOP control.

The encrypted POP algorithm works as follows (as described in [8]):

1. The server generates the POP Proof Value and associates it with the request. This value is typically derived from a random value to protect against replay.
2. The server returns the Encrypted POP Control to the client with the following fields set:
    - request - the original certification request (e.g. a PKCS10 request)
    - cms - EnvelopedData, the encapsulated content type being id-data and the content being the POP Proof Value
    - thePOPAlgID - identifies the algorithm to be used in computing the return POP value
    - witnessAlgID - identifies the hash algorithm used on POP Proof Value to create the field witness,
    - witness - the hashed value of POP Proof Value.
3. The client (using its private key) decrypts the cms field to obtain the POP Proof Value and verifies it by computing a hash of this (using the witnessAlgID) and comparing it against the witness value
4. The client creates the Decrypted POP control as part of a new PKIData. The fields in the DecryptedPOP are:
    - bodyPartID - refers to the certificationRequest in the new PKI request
    - thePOPAlgID - identifies the algorithm to be used in computing the return POP value,
    - thePOP - contains the possession proof
5. The client resubmits the full PKI request, this time including the DecryptedPOP control
6. The server then recomputes the POP value and compares it to the value of the POP. If the values do not match, the server will not issue the certificate. Otherwise, the server issues the certificate and returns it in a full PKI response message.

### 5.1.3 POP Workaround Using Throw-away AIK

Because of TPM-enforced privEK usage restrictions, it is not possible to implement EK POP directly. However, a workaround is possible, utilizing steps very similar to those utilized in the AIK enrollment protocol [4]. Note that our objective here is markedly different than in the AIK enrollment case. In that case, the objectives are to prove that (1) the AIK is TPM-resident, and (2) that the entity requesting the AIK certificate has access to the associated privEK. In this case, we only want to prove that the requesting entity has access to the privEK, so our approach can be simplified as follows:

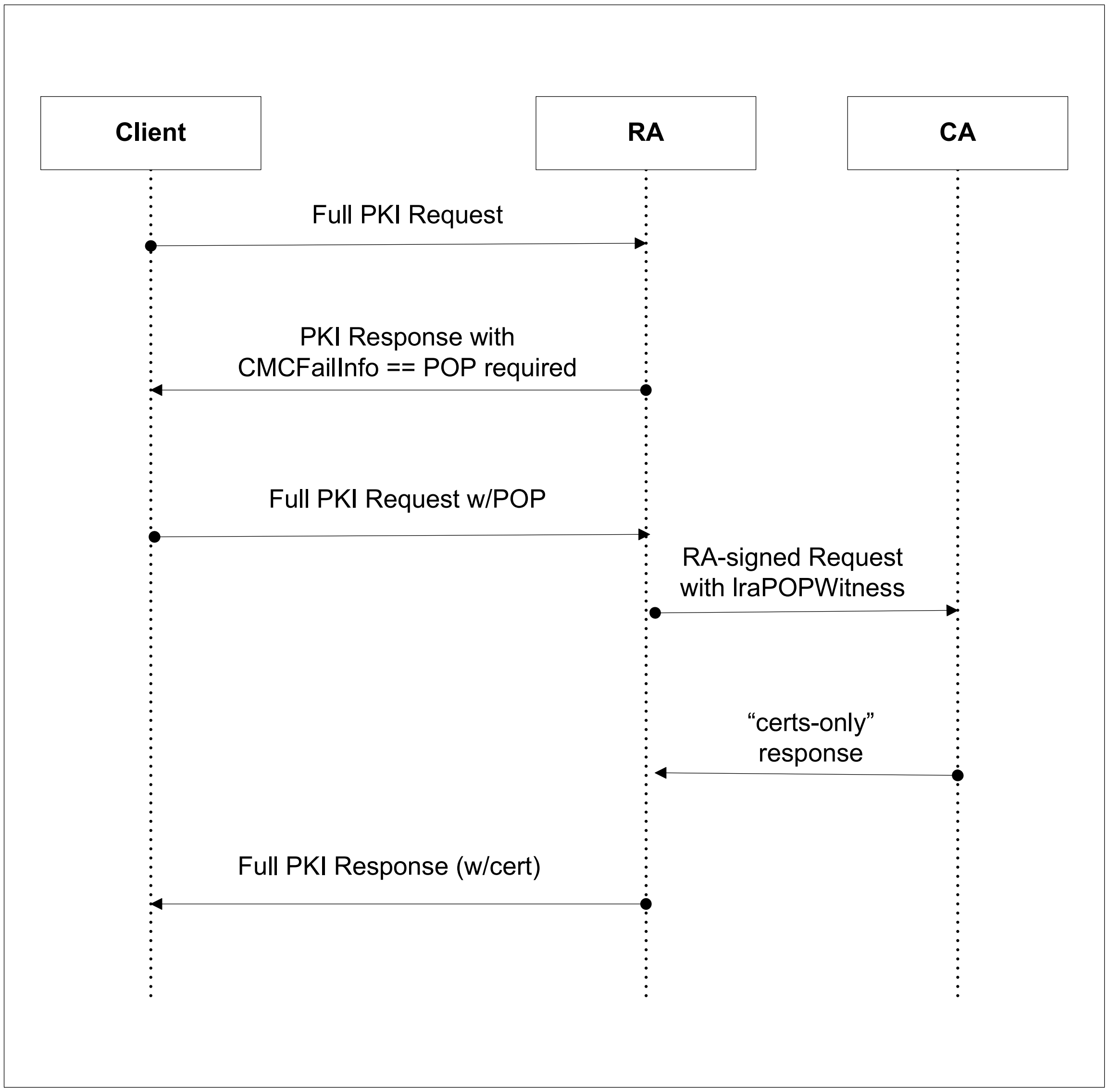
- Enrollment agent creates an AIK pair in the subject TPM
    - The platform (or application software on the platform) issues to the TSS the ***CollateIdentityRequest*** command. In turn the TSS issues the ***MakeIdentity*** command to the TPM. This results in the TPM generating a fresh AIK public key pair.
    - Within the ***MakeIdentity*** function the TPM creates the **IDENTITY_CONTENTS** structure containing the following items: (i) The structure version, (ii) TPM command ordinal, (iii) PrivCADigest label and (iv) AIK_pub_key.
    - The TPM signs **IDENTITY_CONTENTS** structure using the AIK_priv_key, with the resulting signature portion being referred to as the ***identityBinding***.
    - The TPM outputs two (2) items as a result of the ***MakeIdentity*** command: AIK_pub_key and the ***identityBinding***.

- Enrollment agent discards the IdentityBinding structure, but includes the associated AIK_pub_key in the enrollment request to the RA
- The RA creates the TPM_EK_BLOB, which contains the following:
    - HASH(AIK_pub_key)
    - POP Proof value
- The RA encrypts the TPM_EK_BLOB with the EK_pub_key
- This value is returned to the enrollment agent as the POP challenge
- The enrollment agent, upon receiving the TPM_EK_BLOB, issues the TPM_ActivateIdentity command. The TPM will return the POP Proof value, which is then included in the subsequent enrollment request.

This specification makes no recommendation regarding disposition of this particular AIK once EK enrollment is complete. It may be discarded upon completion of this transaction, or it may be saved for later use. Developers are cautioned to carefully consider the security implications of keeping this key for later use.

### 5.1.4 Putting it together: CMC Enrollment with POP

Following is a high-level walk through of CMC enrollment implementing POP:


Client
RA
CA
Full PKI Request
PKI Response with CMCFailInfo == POP required
Full PKI Request w/POP
RA-signed Request with IraPOPWitness
“certs-only” response
Full PKI Response (w/cert)

In this scenario, the client forwards the Full PKI Request to the RA. Keep in mind that the RA may co-reside with the CA in a single platform, and may simply represent a logical module of the CA application. Alternatively, it may reside on a physically separate system. Furthermore, there may be additional RA’s situated between the one illustrated here and the CA, and perhaps even between the one pictured here and the client, but for simplicity, we ignore this.

In any event, the RA pictured above rejects the client’s first attempt, notifying the client that POP is required. The client decodes the POP challenge (and locally validates it), and then resubmits the request with the POP proof. The RA validates the POP proof, and then forwards the request to the CA. The CA returns the certificate(s) to the RA, and the RA forwards the certificate(s) (and any additional data and/or CMC controls) to the client.

## 5.2 CMC Authentication Considerations

CMC supports two methods for integrity verification and data origin authentication: use of a Message Authentication Code (MAC) computed based on a shared secret, and use of the CMS SignedData encapsulation. In many cases, platforms enrolling for EK/Platform certificates will not have a signing key suitable for use in authenticating the enrollment transactions, so we must support platform authentication based on a pre-provisioned shared-secret. In such cases, while using the shared key symmetrically might be simplest, it is significantly more secure if the RA authenticates using a certified signing key, so that is the approach specified here.

In cases where the platform has no suitable public key for authentication (the default enrollment scenario in this specification), asymmetric authentication is used. That is, PKI requests MUST be wrapped in a CMS AuthenticatedData (authenticated with the shared secret), and PKI responses MUST be wrapped in a CMS SignedData (signed by the RA). In cases where the platform has a suitable public key, CMS SignedData MAY be used for PKI requests.

When a shared secret is used for platform authentication, it is REQUIRED that the enrolling platform be securely provisioned out of band with the shared secret. It is RECOMMENDED that the enrolling platform also be securely provisioned with all trust anchors and keys needed to complete the enrollment transaction with the RA and CA (described more fully below) prior to commencement of the enrollment process. If provisioning the trust anchors along with the shared secret is problematic, then it is RECOMMENDED that the trust anchors be retrieved from the RA, in an exchange which is authenticated by the shared secret. The precise manner in which this is accomplished is out of scope for this specification, and implementers are cautioned to recognize that the relative security of the entire enrollment process depends on secure implementation of this procedure.

## 5.3 Implementation Preliminaries

While section 5.1 outlines a CMC-based enrollment process from a high level, there are a number of important nuances and subtleties that must be addressed in order to properly implement EK/Platform enrollment using CMC:

1. The CA will sign the EK/Platform certificates, implying existence of a CA signing key; however, the RA must be able to decrypt the key (K1) used to encrypt the CMS EnvelopedData. A signing key MUST NOT be used for this purpose. Hence, the RA MUST have its own (encryption-only) key for this purpose.
2. If the RA is distinct from the CA, then the RA request on behalf of the client MAY need to be signed by the RA, i.e. the RA will require a signing key which the CA trusts. Also, as noted above, the RA requires a signing key in order to support CMS SignedData encapsulation for data sent from the RA to the client (and this MAY be the same key used to sign requests for the CA).
3. Since CMC requires that Full PKI Requests be encapsulated in either a SignedData or AuthenticatedData, either an acceptable signing key or a shared secret is required in order for the platform to initiate a CMC enrollment transaction. This specification has provisions for both, but support for a shared secret MUST be provided, while support for platform signing keys MAY be provided. This is discussed further below.
4. In order to complete the transaction, the platform must have the RA's (public) encryption key (to encrypt K1, the key used to protect the CMS EnvelopedData), and must also have the RA's public signing key in order to authenticate the CMS SignedData encapsulation. This specification assumes that these keys are provisioned along with trust anchors prior to commencement of EK/Platform enrollment.

For EK/Platform enrollment, we must support the following use cases:

- **UC.enroll.1**: the platform is pre-provisioned with a shared secret and the required trust anchors and RA public keys.
- **UC.enroll.2:** the platform has a valid signing key that the RA will recognize, and has been provisioned with the required trust anchors and RA public keys.

Support for other approaches is out of scope.

### 5.3.1 Data Security and Encapsulation

In order to meet the confidentiality requirements outlined in previous sections, this specification requires the use of CMS EnvelopedData as defined in [8]. We refer to the content encryption key as K1. Briefly, here is an overview of the EnvelopedData encapsulation, along with some implementation-related commentary:

```
EnvelopedData ::= SEQUENCE {
     version CMSVersion,
     originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
     recipientInfos RecipientInfos,
     encryptedContentInfo EncryptedContentInfo,
     unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }
```

In general, OriginatorInfo is optional, and may contain certificates and/or CRLs. For this specification, OriginatorInfo MUST NOT be present.

RecipientInfo serves to relate the manner in which key distribution occurs to the identity of the receiver. There are 4 variants defined. For our purposes, the platform will generate K1 and wrap it with the public key of the RA. For this purpose, we use the KeyTransRecipientInfo variant, defined as follows:

```
KeyTransRecipientInfo ::= SEQUENCE {
     version CMSVersion,  -- always set to 0 or 2
     rid RecipientIdentifier,
     keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
     encryptedKey EncryptedKey }
```

This specification requires that KeyTransRecipientInfo MUST be present, and that the CMSVersion MUST be set to 2. The RecipientIdentifier MUST be populated with the SubjectKeyIdentifier (which will be present in the RA’s encryption key certificate). For KeyEncryptionAlgorithmIdentifier, the RSAES-OAEP [11] algorithm MUST be supported.

The platform is responsible for generating K1. In use cases involving a physical TPM, it is assumed that the platform is capable of producing random numbers of sufficient cryptographic strength for use as K1. In cases involving a virtual TPM, this may or may not be true. In those cases, it is critical that implementers pay special attention to creation of K1. The precise manner of generation does not impact interoperability, so it is not specified here. However, implementers are cautioned to note that predictable keys may result in a compromise of the enrollment protocol.

Since K1 will be also be used for securing packets from the RA to the platform (which may not have a public encryption key), we need some alternative for RecipientInfo when sent from the RA to the platform. That is, in some supported use cases, the platform has no existing public/private key pair that can be used in the enrollment process. This means the platform will have no means for unwrapping an encrypted content key.

Since the platform already knows the value of K1, there is no need to transmit the key, but CMS requires that this structure be present in the message – therefore, to comply with CMC, and to ensure that no unauthorized data is inadvertently leaked via this channel, we simply included the same RecipientInfo in the RA-to-platform messages that the platform previously constructed and sent to the RA. The platform SHOULD verify that the packet from the CA contains the same value as was sent in the enrollment request, either by maintaining a local copy of the encrypted K1, or by re-computing this value upon receipt.

```
    EncryptedContentInfo ::= SEQUENCE {
     contentType ContentType,
     contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
     encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }
```

The following ContentEncryptionAlgorithmIdentifiers MUST be supported:

```
id-aes128-CBC OBJECT IDENTIFIER ::= { aes 2 }
id-aes192-CBC OBJECT IDENTIFIER ::= { aes 22 }
id-aes256-CBC OBJECT IDENTIFIER ::= { aes 42 }
```

Addtitional ContentEncryptionAlgorithmIdentifiers MAY be supported. The CA defines any additional algorithms it supports by indicating supported algorithms in its Certification Practices Statement (CPS).

## 5.4 The CMC Implementation

Here we assume that a shared secret has been provisioned to both the RA and the platform, and that the platform has any necessary trust anchors installed. Further, we assume the RA has both an encryption and a signing key, and that the CA has only a signing key, and that the platform is in possession of the RA signing and encryption keys. Following is a brief synopsis of how enrollment proceeds:

- Platform constructs CMC Full PKI Request, sends to RA/CA
- If EK POP is desired
  - RA constructs unique POP value, sends CMC Full PKI Response message containing POP challenge which is encrypted with the PubEK
  - Platform decrypts POP challenge with PrivEK, validates it, adds proof to CMC Full PKI Request and resubmits to RA/CA
- RA/CA validates platform request, issues certificate(s), returns it/them to the platform in a CMC Full PKI Response

A more detailed description of each step in the enrollment protocol is covered in the following sections.

### 5.4.1 Creation of the Initial Full PKI Request

Following is an illustration of the initial Full PKI Request for EK/Platform enrollment:

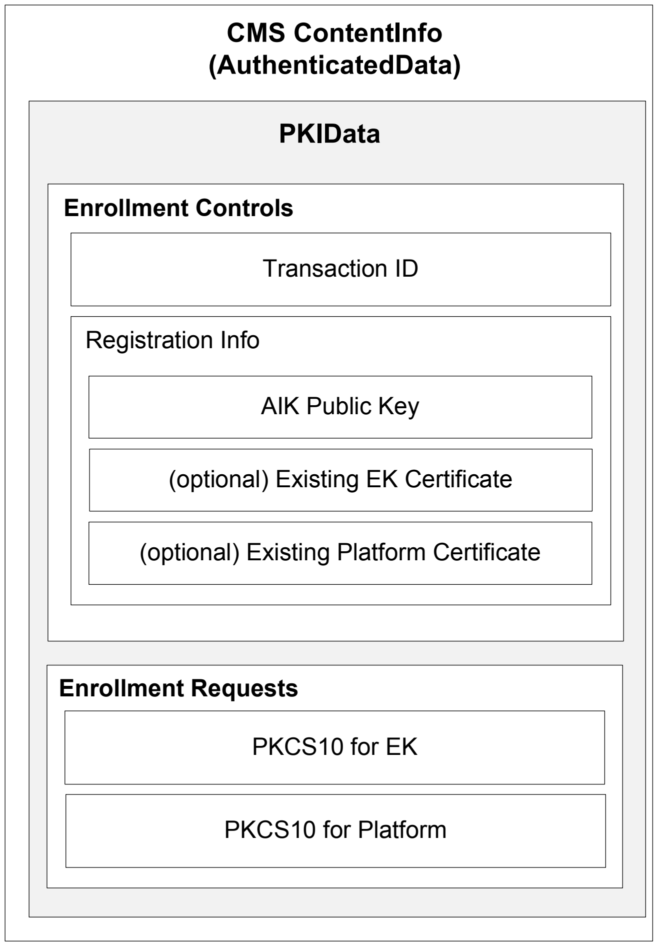


**Figure 14 – Full PKI Request**

This encapsulation is constructed according to the recommendations in [7]. Note that the innermost structure is a PKIData. This is encapsulated in either CMS AuthenticatedData or SignedData, which is in turn encapsulated in EnvelopedData, with the entire bundle again encapsulated in AuthenticatedData or SignedData. In the following sub-sections, we describe the content of the PKIData structure.

#### 5.4.1.1 Transaction ID

This integer value is generated by the platform as a transaction reference value which allows the platform to correlate replies with requests. It is not interpreted by the RA or CA. The platform MAY choose any value which is convenient.

#### 5.4.1.2 Registration Info

The contents of the Registration Info vary according to the type of enrollment that is occurring. If enrolling for a new EK Certificate, or if simultaneously enrolling for new EK and Platform Certificates, this field may be empty. If enrolling for a replacement EK Certificate, this field MAY contain an existing EK certificate. If enrolling for a Platform Certificate alone (i.e, based on an existing EK Certificate), this field MUST contain the associated EK certificate, and MAY contain an existing Platform Certificate.

Note that in case POP is required, then an AIK MUST be provided by the enrollment agent, and this value will be present in the Registration Info. This reginfo element is defined as an OCTET

STRING, and contains the ASN.1 DER-encoded public AIK. This implies that the enrollment client must know a priori that this is required. However, see section 5.4.2 below for additional information.

#### 5.4.1.3 PKCS10 Request(s)

The CMC reqSequence MUST be present. It MUST include the EK and/or Platform Certificate requests in PKCS10 format. Since the PrivEK cannot be used to sign these requests, the signature element MUST hold id-alg-noSignature {id-pkix id-alg(6) 2} in place of the signature. For more information on use of id-alg-noSignature, see [7].

### 5.4.2 Creation of the Full PKI Response Without POP

When the RA receives the FULL PKI request, it MUST authenticate the message. For a detailed discussion, see [8]. Following this, the RA MUST unwrap the EnvelopedData, and authenticate the enclosed PKIData.

If message authentication fails, the RA response is a matter of policy. In environments in which denial of service is not a concern, the RA SHOULD send a full PKI response. Otherwise, the platform may continue to send erroneous requests. In this case, the response SHOULD be as defined below for unrecognized encryption algorithm identifier, except that the CMCFailInfo for this case is authDataFail (13).

Assuming the message validation succeeds, the RA must attempt to unwrap the EnvelopedData. If the ContentEncryptionAlgorithmIdentifier is not supported/recognized, the RA MUST construct a Full PKI Response, and this MUST be authenticated with the RA's signing key. In the controls section, the RA MUST encode the following controls (as described in section 3.2.1.3.4 of [6]):

- Extended CMC Status Info Control (id-cmc-statusInfoV2) with the following fields:
    - Status: failed (2)
    - otherInfo: CMCFailInfo
    - CMCFailInfo: badMessageCheck (1)

Assuming the message is successfully decrypted, the request is signed by the RA and forwarded to the CA. The CA validates the RA signature on the request, fills in any additional certificate fields and/or adds any required extensions based on its policy (NOTE: RA could do this as well), and creates/signs the resulting certificate. This is returned to the RA.

Alternatively, the RA may be implemented as a module of CA. In this case, the PKCS10 requests MAY be passed directly to the CA, with no further encapsulation, RA signature, etc. That is, the RA MAY encapsulate all knowledge of the CMC protocol API, and the CA MAY trust the RA implicitly. This is an implementation detail which falls beyond the scope of this specification.

Regardless of which approach is taken, the CA will produce either the EK/Platform certificate(s), or an error message. The RA, upon receiving the response from the CA, MUST create a Full PKI response which will be returned to the platform. To maintain confidentiality, the RA MUST encapsulate the PKIData in a CMS EnvelopedData and encrypts this with K1.

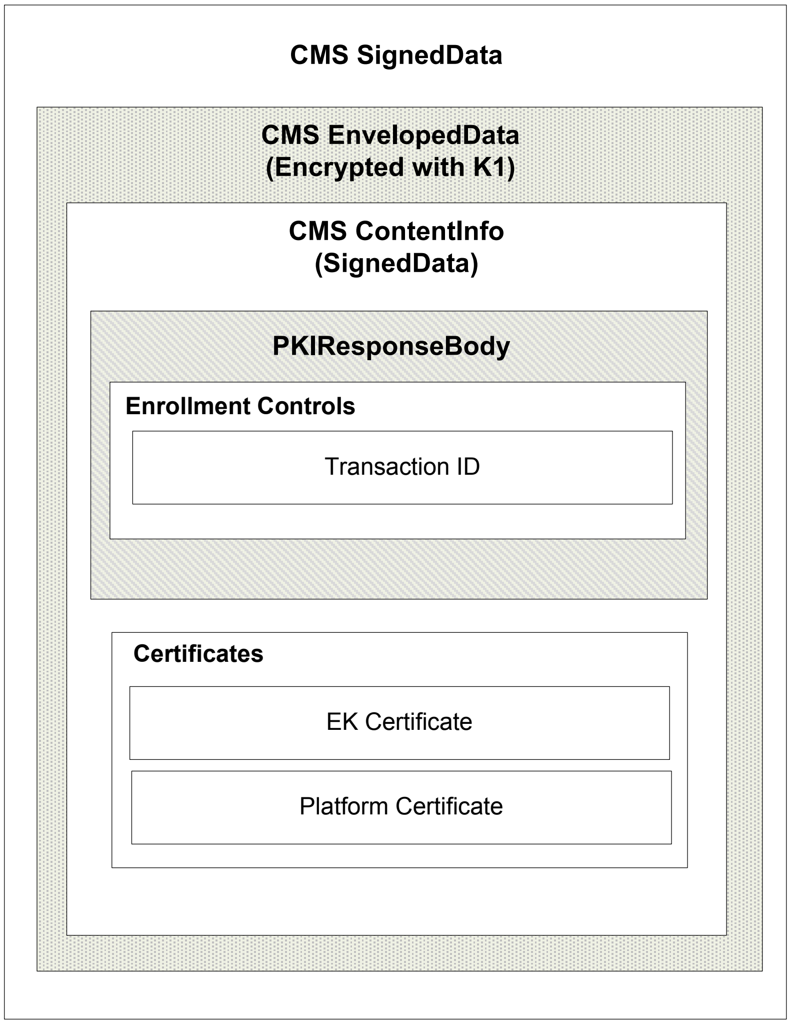


**Figure 15 - Full PKI Response**

### 5.4.3 Creation of the Initial Full PKI Response With POP

When the RA receives the PKI request, it MUST authenticate the message. For a detailed discussion, see [6]. Following this, the RA MUST unwrap the EnvelopedData, and authenticate the enclosed PKIData.

If message authentication fails, the RA response is a matter of policy. In environments in which denial of service is not a concern, the RA SHOULD send a response. In this case, the response SHOULD be as defined below for unrecognized encryption algorithm identifier, except that the CMCFailInfo for this case is authDataFail (13).

Assuming the message validation succeeds, the RA must attempt to unwrap the EnvelopedData. If the ContentEncryptionAlgorithmIdentifier is not supported/recognized, the RA MUST construct a Full PKI Response, and this MUST be authenticated with the RA's signing key. In the controls section, the RA MUST encode the following controls (as described in section 3.2.1.3.4 of [6]):

- Extended CMC Status Info Control (id-cmc-statusInfoV2) with the following fields:
    - Status: failed (2)

  - otherInfo: CMCFailInfo
  - CMCFailInfo: badMessageCheck (1)

Assuming message decryption succeeds, the RA examines the enrollment controls section of the request. Finding no decrypted POP control in the request, the RA infers that this is the first message. The RA extracts the EK public key from the enclosed EK CSR, and extracts the AIK from the Registration Info. If any error is encountered in parsing the enrollment request, the RA constructs and sends a Full PKI Response, which will be authenticated with the RA's signing key. In the controls section, the RA MUST encode the CMC Status Info Control as in the example above. The RA MAY send the value "badRequest" with no further information about the failure, but if possible, the CMCFailInfo value which most closely identifies the problem SHOULD be sent. The RA MAY also include ExtendedFailureInfo to further identify the problem.

If POP is required, but the enrollment agent did not include an AIK, the RA MUST construct a Full PKI Response, and this MUST be authenticated with the RA's signing key. In the controls section, the RA MUST encode the following controls (as described in section 3.2.1.3.4 of [6]):

- Extended CMC Status Info Control (id-cmc-statusInfoV2) with the following fields:
  - Status: failed (2)
  - otherInfo: CMCFailInfo
  - CMCFailInfo: popFailed (9)

The enrollment agent, upon receiving the popFailed error, may infer that POP is required, and resubmit the request, this time including the AIK.

If the message passes all RA checks, then the RA must now construct the POP challenge value (R), which will be encrypted, and only accessible to the enrollee if it has the appropriate private key. The manner in which the challenge value is constructed is implementation-dependent, and out of scope for this specification. The purpose of the challenge value is to ensure, to a high degree of probability, that the platform has the private EK. The primary threats to this mechanism would consist in a malicious platform being able to predict the value of R somehow. This might be accomplished if either the method of generating R is weak, or if an R value from a previous protocol run (for which the platform knows the correct value) could be re-used in a later run. Hence, construction of R MUST ensure that

- It is not practical to guess it in advance
- The odds of colliding R values is vanishingly small (minimizing likelihood of re-usability, given a valid, earlier value).

If the RA wishes to remain stateless, then it MAY implement a challenge generation mechanism that depends on the enrolling platform somehow, and which allows stateless reconstruction of the challenge when the platform submits a challenge response. Alternatively, the RA MAY choose to maintain a record of outstanding challenges. This is implementation dependent.

Regardless of how the RA generates the challenge R, the RA constructs the encrypted TPM_EK_BLOB as follows:

$$\{\mathrm{Hash}(\mathrm{PubAIK}) \mid R\}_{EK\_pub}$$

Note that while the general definition for the TPM_EK_BLOB includes optional PCR information, we do not permit such optional PCR information here.

Now, the RA constructs a Full PKI Response, which will be authenticated with the RA's signing key. In the controls section, the RA MUST encode the following controls:

- Extended CMC Status Info Control (id-cmc-statusInfoV2) with the following fields:

  - Status: failed (2)
  - otherInfo: CMCFailInfo
    - CMCFailInfo: popRequired (8)
- The encrypted POP control (described below)

The RA must construct the Encrypted POP control, which has the following ASN.1 definition:

```
     EncryptedPOP ::= SEQUENCE {
           request          TaggedRequest,
           cms              ContentInfo,
           thePOPAlgID      AlgorithmIdentifier,
           witnessAlgID     AlgorithmIdentifier,
           witness          OCTET STRING
      }
```

As noted above, the POP proof value is R. The RA constructs the Encrypted POP control, where the following fields MUST be set as specified below:

- request – contains the original PKCS10-encoded AIK certificate request from the platform
- cms – EnvelopedData with a content type of id-data and the content being $\{TPM\_EK\_BLOB\}_{EK\_pub}$
- thePOPAlgID – identifies the encryption algorithm for encrypting R prior to returning it in the follow up request
- witnessAlgID – identifies the hash algorithm to be used in computing the return POP value
- witness – the hashed value of R

The PKI response is then returned to the platform. Note that the RA/CA need not maintain local state; this is an implementation decision.

### 5.4.4 Creation of the Full PKI Request Including POP

Upon receiving the initial PKI Response, the platform first validates the message using the shared secret. Invalid messages MUST be discarded, and SHOULD be logged.

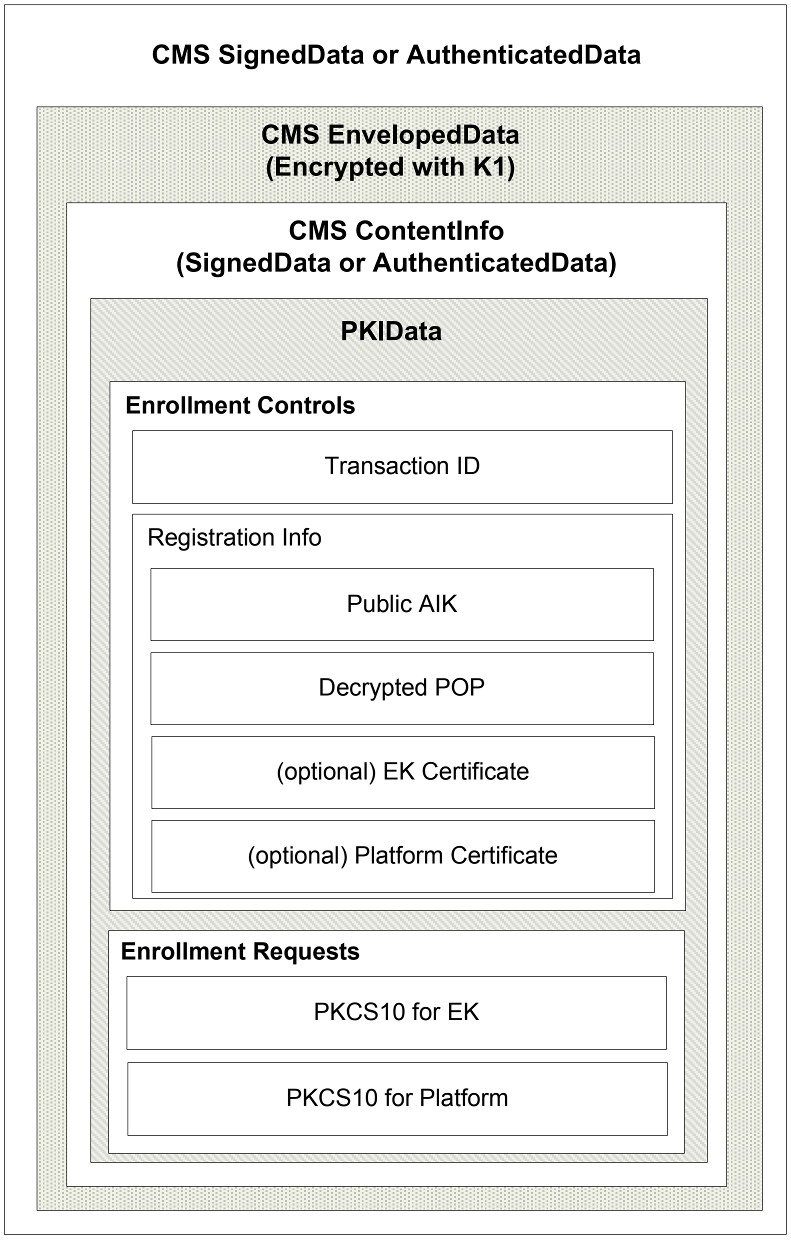
Assuming the message validation succeeds, the platform then extracts the encrypted POP control, yielding $\{TPM\_EK\_BLOB\}_{EK\_pub}$.

This value is then decrypted through a call to the TSS Activate Identity function. Following this call, the TSS will return R. The platform, by applying to R the hash algorithm defined by thePOPAlgId in the encrypted POP control, will derive the witness value. This MUST be compared with the witness value in the encrypted POP control; if these do not match, the transaction has failed, so the platform SHOULD log this event and MUST discard the message.

Assuming the locally computed witness value matches the one contained in the PKI Response message, the platform constructs a new PKI request, this time adding the Decrypted POP control to the enrollment controls. That is, the request is exactly the same as the initial request, except that the Decrypted POP control is added. The Decrypted POP control has the following ASN.1 definition:

```
      DecryptedPOP ::= SEQUENCE {
           bodyPartID       BodyPartID,
           thePOPAlgID      AlgorithmIdentifier,
           thePOP           OCTET STRING
      }
```

The BodyPartID MUST refer to the EK PKCS10 enrollment request contained in the new Full PKI Request message. The AlgorithmIdentifier value MUST be copied directly from the PKI response message. The OCTET STRING (thePOP) MUST contain the transformed R. This message is illustrated below:


CMS SignedData or AuthenticatedData
CMS EnvelopedData (Encrypted with K1)
CMS ContentInfo (SignedData or AuthenticatedData)
PKIData
Enrollment Controls
Transaction ID
Registration Info
Public AIK
Decrypted POP
(optional) EK Certificate
(optional) Platform Certificate
Enrollment Requests
PKCS10 for EK
PKCS10 for Platform

It is possible for an attacker to capture this message and replay it again and again. Rather than adding complexity to the enrollment protocol in an effort to mitigate this, it is RECOMMENDED that the CA implement some appropriate means of anti-replay detection. For example, the CA has considerable latitude in formulation of the POP value. One simple replay mitigation mechanism consists of constructing the POP as a concatenation of R and a timestamp. If a request is received for which the timestamp is considered to be out of date (e.g., it is some number of "ticks" behind the current timestamp), then the request could be discarded. Alternatively, the CA could maintain some state associated with the ongoing POP transaction, and discard this when either the transaction completes, or some time interval elapses. Since this implementation choice need not influence interoperability, we leave the design of this mechanism to the CA implementer.

This updated Full PKI Request is forwarded by the platform to the RA/CA.

### 5.4.5 Creation of the Full Final PKI Response with POP

When the RA receives the FULL PKI request (with POP), it MUST first authenticate the message. For a detailed discussion, see [8]. Following this, the RA MUST unwrap the EnvelopedData, and authenticate the enclosed PKIData.

If message authentication fails, the RA response is a matter of policy. In environments in which denial of service is not a concern, the RA SHOULD send a full PKI response. Otherwise, the platform may continue to send erroneous requests. In this case, the response SHOULD be as defined below for unrecognized encryption algorithm identifier, except that the CMCFailInfo for this case is authDataFail (13).

Assuming the message validation succeeds, the RA must attempt to unwrap the EnvelopedData. If the ContentEncryptionAlgorithmIdentifier is not supported/recognized, the RA MUST construct a Full PKI Response, and this MUST be authenticated with the RA's signing key. In the controls section, the RA MUST encode the following controls (as described in section 3.2.1.3.4 of [6]):

- Extended CMC Status Info Control (id-cmc-statusInfoV2) with the following fields:
  - Status: failed (2)
  - otherInfo: CMCFailInfo
  - CMCFailInfo: badMessageCheck (1)

Assuming the message is successfully decrypted, the RA examines the controls, and finding the Decrypted POP control, extracts the PubAIK from the IDENTITY_PROOF structure. Using this, the RA re-computes R as previously specified above.

Using R, the RA computes the transformation by applying the algorithm specified by thePOPAlgID to R, and then compares this value to thePOP (contained in the Decrypted POP control). If these values match, the platform has proved possession of the PrivEK. If these do not match, the request MUST be discarded, and SHOULD be logged.

The RA must now transform the request into one that is acceptable to the CA. This is accomplished by marking the controls which have already been processed, adding the lraPOPWitness control which encapsulates the platform's PKCS10 request, and signing this with the RA signing key. For a detailed explanation of this process, see [6].

The resulting, modified request is forwarded to the CA. The CA validates the RA signature on the request, fills in any additional certificate fields and/or adds any required extensions based on its policy (NOTE: RA could do this as well), and creates/signs the resulting certificate. This is returned to the RA.

Alternatively, the RA may be implemented as a module of CA. In this case, the PKCS10 request(s) MAY be passed directly to the CA, with no further encapsulation, RA signature, etc. That is, the RA MAY encapsulate all knowledge of the CMC protocol API, and the CA MAY trust the RA implicitly. This is an implementation detail which falls beyond the scope of this specification.

Regardless of which approach is taken, the CA will produce either the requested certificate(s), or an error message. The RA, upon receiving the response from the CA, MUST create a Full PKI response which will be returned to the platform. To maintain confidentiality, the RA MUST encapsulate the PKIData in a CMS EnvelopedData and encrypts this with K1. K1, in turn, is encrypted with the PubEK extracted from the request, and the entire response is wrapped in CMS

AuthenticatedData, and signed with the public key of the RA. This message is returned to the platform.

# 6 Security Considerations

This document describes a protocol for enrolling TPMs for EK and Platform certificates. There are very few applications for these certificates, with the primary standardized use case being AIK enrollment. The basic aim of AIK enrollment is to create AIK certificates in support of two potential uses: platform attestation, and key certification. These things may seem unrelated to EK and Platform certificate enrollment, but in terms of security considerations, it is in the context of these operations that we must begin.

## 6.1 Platform Attestation

For platform attestation to be useful, it must be trustworthy. In order for it to be trustworthy, we must have good reasons for believing the associated assertions. Ultimately, we derive this assurance via a complex transitive trust chain that begins with the AIK, continues with the platform and EK certificates, and is anchored by the TPM. For this to work, we have to believe that the platform contains a TPM that has a specific set of well-defined properties [16].

How do we know that the TPM is genuine? The only way we know this is if (1) we are assured that a particular public key is indeed a pubEK (that, by definition, resides in a genuine TPM), and (2) the entity can prove that it controls the privEK associated with the pubEK. In the case where our assurance derives from an EK certificate, then underlying our assurance is an article of faith, a belief that the issuer of the EK certificate would not knowingly lie to us, and is diligent in protecting its own private key, as well as in verifying the integrity of the TPM and the surrounding manufacturing, provisioning, and enrollment processes prior to issuing the certificate.

In the process of discussing the anchor to which the trust chain is bound, we referenced a critical link in the chain: the Platform certificate. In order to believe that not only is there a genuine TPM, but that it resides in a platform that has a specific set of well-defined properties, we again must rely on an article of faith, this time based on an EK certificate *as well as* a Platform certificate that references that same EK. And in this case, we must believe that both issuers are trustworthy and diligent.

Then, with those links in place, we add another: the AIK. In order for attestation to be useful, we must rely only on the AIK certificate – we are not allowed to see the EK or Platform certificates. So, if we are to accept that a key truly is an AIK, we must believe that the Attestation CA (ACA) is trustworthy, that it diligently validated the both the EK and Platform certificates, and that it is faithfully reporting the platform qualities and attributes.

While this all sounds rather tenuous, and it certainly depends heavily on “trust”, the fact is that with the appropriate level of care and investment, it is possible to make the various steps in these processes reliable. But if any of the related steps are *not* reliable, everything that comes after is suspect. And that is a very difficult problem to solve, given the realities of global equipment manufacturing.

Clearly, there is a lot at stake here. If an EK certificate is issued carelessly, for example, to an attacker who controls the associated private key, the attacker has the first piece of the puzzle, arguably the foundational element of a potentially devastating attack. And if such an attacker can obtain a platform certificate for this bogus TPM, he is now in a position to obtain an AIK certificate. And with this AIK certificate, an attacker can produce false attestations that allow him to potentially undermine whatever ecosystem the associated device participates in.

## 6.2 Certified Keys

The other AIK use case mentioned above involves using the AIK to “prove” that another key is TPM-resident. Such keys are “certified” by signing what amounts to an assertion that the key is TPM-resident using the AIK. Due to TPM-enforced restrictions on AIK usage, we can believe the veracity of such assertions if the signing key is truly an AIK, but proof for this this leads us into the chain of transitive trust described above. If any of the links in that chain are weak, this directly impacts our level of assurance.

Typically, certified keys might be used for device authentication, to “prove” that the entity wielding the key is a particular device. If this key can be used by a malicious system that is not the represented device, this has the potential to undermine whatever system the certified key is a part of. Again, such attacks are potentially devastating, so the integrity of the EK and Platform certificate enrollment processes is fundamental to the integrity of the certified key enrollment process.

## 6.3 Threats to Physical TPM Enrollment

It is within the context outlined above that we must consider EK and Platform certificate enrollment. Now, this particular specification covers *many* different use cases, and it would be quite complex to try to unravel each and every one of those, and to anticipate all of the associated threats. Further, given that the use cases given in this document are very general, and are based on what we may envision, as opposed to what has actually been implemented, it may be more productive to discuss general threats in terms of phases in the deployment lifecycle.

Recall from section 2.3 that we discussed 3 phases:

- Pre-deployment, covering various stages of manufacturing
- Deployment, cover post-manufacturing
- Retirement/redeployment

If we think of a timeline, stretching from TPM design and fabrication, through platform design and manufacturing, through deployment, and on to platform retirement or redeployment, we may note several things:

- During TPM design/fabrication
    - If TPM design is not strictly controlled and validated, we may have a failure at the very beginning
    - If TPM fabrication is not strictly controlled, rogue functionality that is not part of the “authorized” design may be introduced
    - If storage of finished inventory is not strictly controlled, counterfeit TPMs may be introduced
- During delivery to the platform manufacturer, and up until platform fabrication
    - If inventory and delivery is not strictly controlled, counterfeit TPMs may be introduced, or existing TPMs may be modified
- During platform design
    - If platform design is not strictly controlled and validated, rogue elements may be introduced to the platform (elements which are not part of the “authorized” design)
- During platform manufacture
    - If inventory and delivery are not strictly controlled, malicious parts may be introduced into the platform (parts which appear or are thought to be well understood, but which are not)
    - If manufacturing processes are not carefully validated, what is being manufactured may not match what was designed (rogue elements may be

        introduced, things may not be connected, “wired” in the manner intended,etc.)

  - During platform delivery
    - If inventory and delivery are not strictly controlled, platforms may be maliciously modified/substituted
  - During platform re-deployment
    - Depending on the circumstances of the deployment, platforms may be maliciously modified at some point in the lifecycle

Looking at the list above, we may note that early enrollment potentially mitigates a number of threats. For example, if we assume that the TPM manufacturer has adequately secured and validated design and manufacturing processes, then provisioning EKs and issuing EK certificates as part of the TPM manufacturing process makes it *very* difficult to inject counterfeit TPMs further up the line. The attacker would have to subvert the EK CA signing key, or have a method for maliciously modifying devices after manufacture.

Likewise, if we assume that the platform manufacturer has adequately secured and validated design and manufacturing processes, then issuing Platform certificates as part of the platform manufacturing process makes it more difficult to inject counterfeit platforms further up the deployment lifecycle timeline. Note that in this case, an attacker could still maliciously modify existing platforms (including removal of TPMs for use in rogue platforms), so even at this early stage, our returns are diminishing.

However, now look at the situation in a deployment use case scenario. For example, let’s take a typical corporate enterprise. Suppose the IT department receives a shipment of laptops. Someone in the IT department initializes a laptop, and runs a program that takes ownership of the TPM, generates an EK, uses that EK to enroll for EK and platform certificates using a local CA that implements the enrollment protocol described above.

Think about the various leaps of faith required: you have to believe that the TPM is “genuine”. You have to believe that the platform is “trustworthy”, i.e. that when you issue TPM commands and receive responses, that you are actually interacting **directly** with the TPM, and that there are no rogue parts in the platform (e.g. a subverted south bridge chip) that are “enhancing” your experience. You have to believe that the TPM conforms to all of your expectations. That’s a *lot* of leaps.

In general, we can observe two things here:

- Assuming high-assurance pre-manufacturing processes, then as temporal distance from point of manufacture increases, assurance decreases
- As system complexity increases (e.g., as we go from TPM, one part, to platform, many parts), assurance again decreases markedly.

The bottom line is that there are many potential threats to these processes, and if you intend to implement EK and/or platform certificate enrollment, you should carefully evaluate what you have to lose if anything goes wrong, and who might be motivated to attack you, and then evaluate your provisioning chain accordingly.

## 6.4 Threats to Virtual TPM Enrollment

Virtual TPMs are markedly different from physical TPMs. For one thing, we *expect* them to materialize late in the platform lifecycle, and we may even expect them to have a relatively short lifetime, depending on the environment. It is important to note that we can depend on an underlying physical TPM as a root of trust, but then again, our previous discussion of threats to the physical TPM may lead one to conclude that we may be building a proverbial house of cards here.

It is possible that someone would want to provision EK and platform certificates to a vTPM that does not rely on an underlying physical TPM, but it is hard to imagine why. In such cases, everything would depend on the environment in which enrollment occurs, the security controls around the enrollment agent, the execution environment of the vTPM, and many other factors. However, without a concrete real-world scenario to discuss, it is impossible to articulate a rational set of threats. So, we set this case aside, and only describe those cases that do rely on a physical TPM.

The primary threats to vTPM enrollment are straightforward:

- Attacker wishes to obtain EK certificate for an unauthorized key. That is, the attacker wishes to impersonate a vTPM.
- Attacker wishes to obtain invalid platform certificate
  - Certificate for platform that is different than the one making the request
  - Certificate containing invalid security qualities/attributes
- Attacker wishes to interfere with EK/Platform certificate enrollment of valid vTPM
  - By impersonating RA/CA
  - By Denial of Service attack on RA/CA

## 6.5 Countermeasures

Any discussion of countermeasures for threats against physical TPMs would be potentially long and varied, and we do not attempt to undertake such a detailed discussion here. Rather, the reader is referred to the current literature on supply chain security. Aside from this, the best countermeasure may be to use a platform for which the EK and Platform certificates are created during manufacturing, and which comes from a manufacturer you have reason to trust.

In terms of countermeasures relating to vTPMs, we described two types of attacks: denial of service, and misrepresentation. Denial of service attacks can be mitigated by various active measures, depending on the environment. In any virtualization environment, there is a provider who presumably has recourse against misbehaving client software, and the provider should be in a position to address such issues. If not, you should probably consider finding a new provider.

For misrepresentation attacks, the best countermeasure is to rely on the VMM for proxy enrollment. If the VMM has a properly initialized and robustly provisioned physical TPM (note all the security concerns above), and the VMM signs enrollment requests with a certified key [14] that is sealed to VMM state, then you have very high assurance relative to enrollment requests.

# 7 References

## 7.1 Normative References

[1] Trusted Computing Group, *IWG Reference Architecture for Interoperability (Part 1)*, Specification Version 1.0, http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_reference_architecture_for_interoperability_specification_part_1_version_10, June 2005.

[2] Trusted Computing Group, IWG Architecture Part II – Integrity Management Version 1.0, http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_architecture_part_ii__integrity_management_version_10, November 2006.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", Internet Engineering Task Force RFC 2119, March 1997.

[4] Trusted Computing Group, *A CMC Profile for AIK Certificate Enrollment*, Specification Version 1.0, Revision 7, September 2011

[5] Trusted Computing Group, *Credential Profiles*, Specification Version 1.1, Revision 1.014, http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_tcg_credential_profiles_specification, May, 2007

[6] Certificate Management Messages over CMS, RFC 5272, ftp://ftp.rfc-editor.org/in-notes/rfc5272.txt, June 2008.

[7] Certificate Management Messages over CMS (CMC): Transport Protocols, RFC 5273, ftp://ftp.rfc-editor.org/in-notes/rfc5273.txt, June 2008

[8] Cryptographic Message Syntax (CMS), RFC 3852, ftp://ftp.rfc-editor.org/in-notes/rfc3852.txt, July 2004

[9] PKCS #10: Certification Request Syntax v1.5, RFC 2314, ftp://ftp.rfc-editor.org/in-notes/rfc2314.txt, Oct 1997

[10] Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax RFC 3565, ftp://ftp.rfc-editor.org/in-notes/rfc3565.txt, July 2003

[11] Use of the RSAES-OAEP Key Transport Algorithm in the Cryptographic Message Syntax RFC 3560, ftp://ftp.rfc-editor.org/in-notes/rfc3560.txt, July 2003

[12] Internet X.509 Public Key Infrastructure, RFC 3280, ftp://ftp.rfc-editor.org/in-notes/rfc3280.txt, April 2002

[13] Internet X.509 Public Key Infrastructure Online Certificate Status Protocol - OCSP RFC 2560, ftp://ftp.rfc-editor.org/in-notes/rfc2560.txt, June 1999

[14] Trusted Computing Group, *Subject Key Attestation Evidence Extension*, Version 1.0, Revision 0.7, June 2005

[15] Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, RFC 3647, ftp://ftp.rfc-editor.org/in-notes/rfc3647.txt, November 2003

[16] Trusted Computing Group, *TCG Main Specification*, Version 1.2, Revision 116, March 2011

[17] Trusted Computing Group, *Virtualized Trusted Platform Architecture Specification*, Version 1.0, Revision 0.26, September 2011

## 7.2 Informative References

[18] Trusted Computing Group, *TNC IF-IMC*, Specification Version 1.1, January 2006.

[19] Trusted Computing Group, *TNC IF-IMV*, Specification Version 1.1, January 2006