TRUSTED® COMPUTING GROUP

GUIDANCE

EK-Based Key Attestation with TPM Firmware Version

Version 1
October 24, 2025

Contact: admin@trustedcomputinggroup.org

TCG Published

DISCLAIMERS, NOTICES, AND LICENSE TERMS

Contents

1 Purpose **5**
1.1 In Scope 5
1.2 Out of Scope 5

2 Introduction **6**

3 Protocol **8**
3.1 Privacy Concerns 9

4 Variant: Firmware Version retrieval independent of AK deployment **10**

References **12**

List of Figures

1 Example TPM Bootloader and Firmware 6
2 Typical AK Attestation Protocol 7
3 Protocol Overview 8
4 Protocol Variant Overview 10

1 Purpose

This document provides guidance to TPM key-attestation verifiers who would like to receive a trustworthy signal of the TPM's current firmware version before trusting the attested key. An example of where this might be useful is if the verifier wants to include the TPM's current firmware version as metadata in an Attestation Key (AK) credential based solely on trust in the TPM's Endorsement Key (EK).

1.1 In Scope

- Devices with a TPM 2.0 and a known decryption (default) EK public key or credential as described in the TCG EK Credential Profile [1]
- Verifiers who trust the EK public keys or the issuers of the EK credentials in the target devices

1.2 Out of Scope

- Details about the nature of the AK credential (e.g., whether it is an ordinary certificate, or some other structure that bestows trust onto an AK)
- The detailed protocol in use dictated by the Verifier (e.g., AK proof-of-possession)
- Data formatting concerns
- Algorithm-specific details of the EK
- Signing EKs
  - Signing EKs can be used directly with `Certify` or `CertifyCreation` instead.
- Devices that do not have a trusted EK public key or credential
  - The protocols in this document depend on a trusted EK.
- Devices that have firmware-limited EK credentials (see "TPM Firmware-Limited and SVN-Limited Objects" in the TPM Library [2] and "Firmware-limited EKs" in the EK Credential Profile [1])
  - The *version* field in the firmware-limited EK credential is a clearer signal about the TPM's current firmware version, so no additional steps are needed in order to ascertain the version.
- When and how the TPM's firmware is updated, along with anti-rollback concerns
- How to identify a particular legitimate TPM as the expected TPM in the expected device
  - The Platform Certificate Profile [3] provides a solution to this concern.

2 Introduction

Version 1.83 of the TPM Library Specification ([2], “TPM Firmware-Limited and SVN-Limited Objects”) describes a motivation for attesting to the current version of the firmware of a TPM. It introduces a simplified model of a TPM as a microcontroller with some application firmware that implements the TPM 2.0 external interface, and a bootloader that verifies and launches the application firmware. Figure 1 depicts this simplified model:

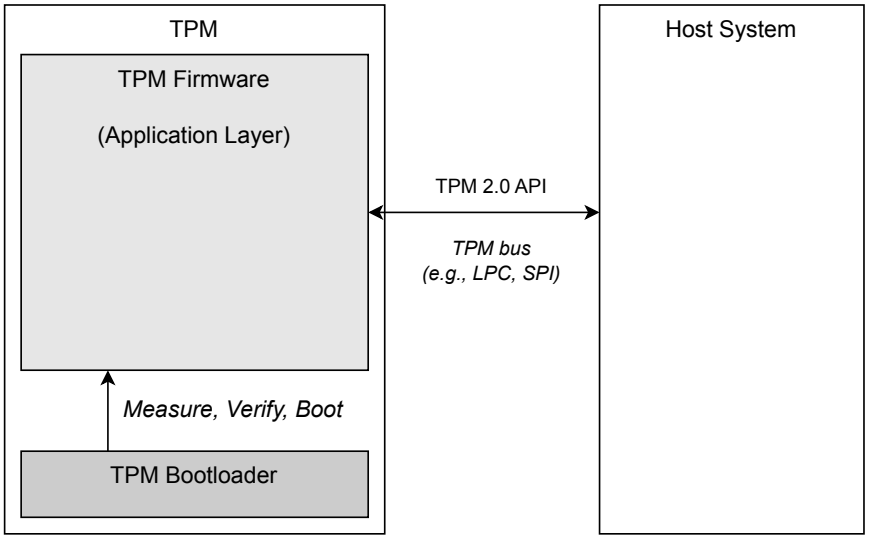

Figure 1: Example TPM Bootloader and Firmware

The ability to attest to the version of the TPM’s firmware can be of interest any time the TPM’s firmware is updated.

Version 1.83 of the TPM Library Specification introduced firmware-limited and SVN (Security Version Number)-limited objects, which provide a strong cryptographic binding between the firmware version of the TPM, and keys wielded by that firmware. A TPM with this capability can have a Firmware-Limited EK Credential (see [1]) that directly vouches for the current version of the TPM’s firmware: if the TPM’s firmware changes, its Firmware-Limited EK is automatically rotated and the new Firmware-Limited EK is certified by the TPM’s bootloader.

For TPMs that do not have this capability (e.g., because they implemented an earlier version of the TPM Library Specification, or because they targeted a platform-specific TPM profile that did not require the feature), it can still be useful to check the firmware version of the TPM when attesting to Attestation Keys using the EK.

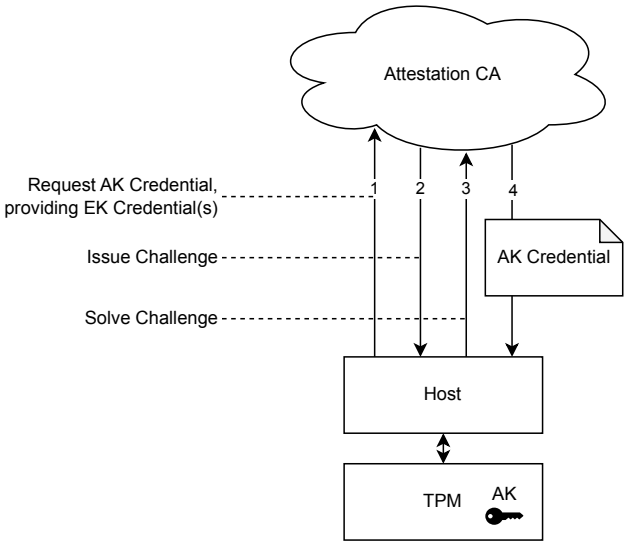

Figure 2: Typical AK Attestation Protocol

Figure 2 depicts a typical AK attestation protocol whereby a host with an attached TPM provides sufficient information for an Attestation CA to issue a credential (e.g., a certificate) to an AK held within its TPM. This AK can then be used for other attestation purposes (e.g., PCR (Platform Configuration Register) quotes).

The typical AK attestation protocol uses the following steps or similar:

1. Host requests AK credential from CA, providing the AK public key and its EK credential(s).
2. CA sends challenge to Host.
3. Host decrypts challenge (`TPM2_ActivateCredential()`) and sends the decrypted challenge back to CA.
4. CA issues AK credential and sends it to Host.

The typical AK attestation protocol leverages `TPM2_ActivateCredential()` to directly challenge a TPM's EK to vouch for a loaded AK, but `ActivateCredential` does not allow also getting the EK to vouch for the TPM's current firmware version.

This document describes an AK attestation protocol based on `TPM2_Import()` and `TPM2_Certify()` that gets a TPM's EK to vouch for a loaded AK along with the TPM's current firmware version. It is designed to use the same number of network hops as the typical `ActivateCredential`-based protocol. As discussed in Clause 3.1, it also supports the same privacy-preserving capability as the typical `ActivateCredential`-based protocol.

Verifiers that adopt this new AK attestation protocol can incorporate the attested TPM firmware version in a number of ways, for example:

- Checking the attested TPM's current firmware version against a list of distrusted firmware versions before issuing the AK credential
- Including the attested TPM's current firmware version as metadata in the AK credential

3 Protocol

In this attestation protocol, the Verifier provisions a restricted HMAC (Hash-based Message Authentication Code) key into the TPM by encrypting it to a trusted EK, then uses the restricted HMAC key to attest to a candidate AK along with the TPM's current firmware version. The attestation carries trustworthy evidence of the TPM's current firmware version, without the Verifier relying on anything but the trustworthiness of the EK. At the end of the protocol, the Verifier confirms that the AK is resident in a legitimate TPM, which is running a known firmware version, before issuing a credential to that AK.

The provisioned key is an HMAC key for these reasons:

- Every TPM 2.0 supports HMAC keys.
- The protocol is inherently symmetric: The Verifier performs both key generation and verification.

The flow is depicted in Figure 3. Reference code for this protocol is available at https://github.com/TrustedComputingGroup/tpm-fw-attestation-reference-code.

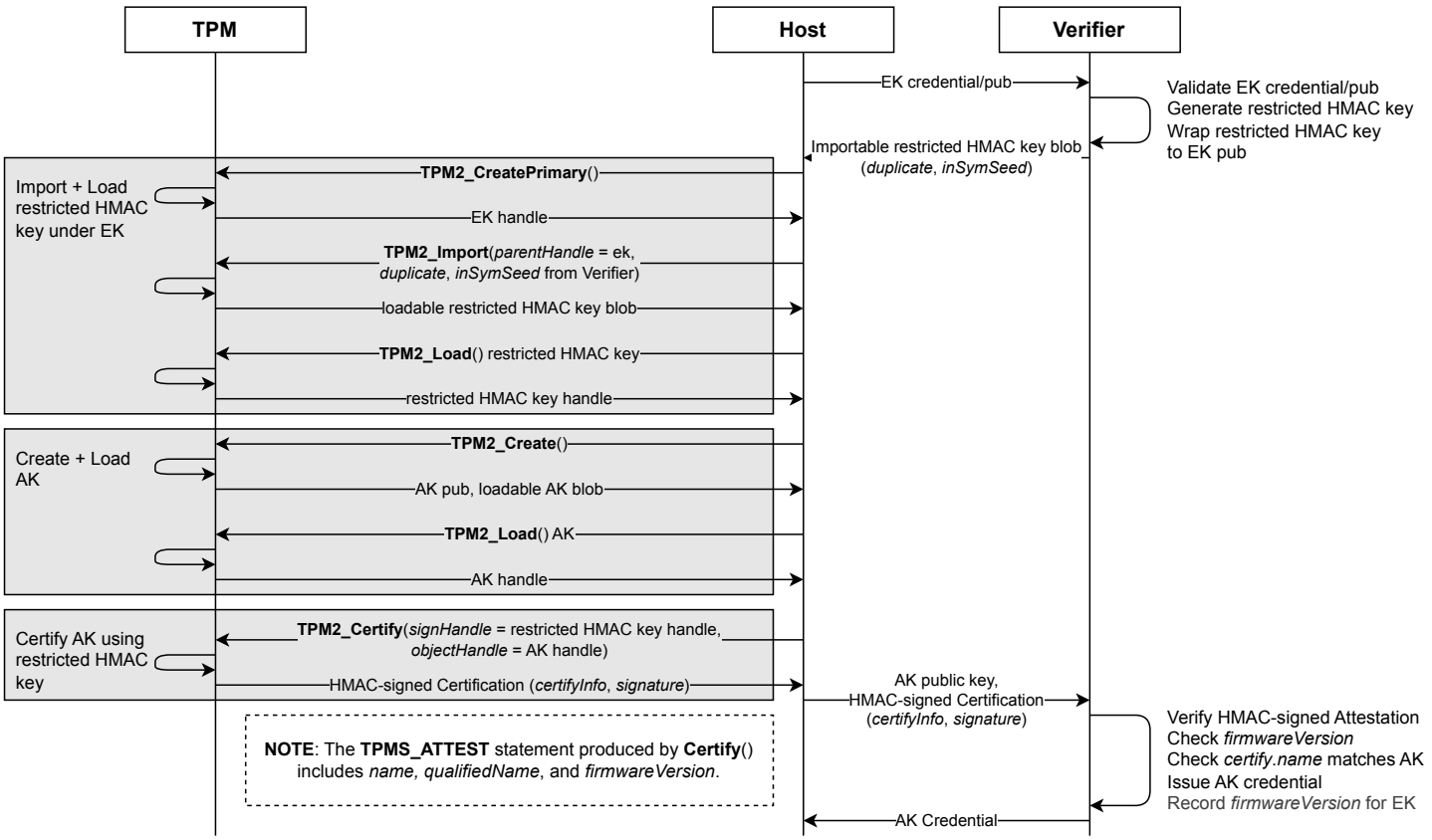

Figure 3: Protocol Overview

1. The Host requests an AK credential from the Verifier, providing the EK public key or credential(s).

   Note:

   In some environments, the Verifier may already have a list of known expected EK public keys.

2. The Verifier checks that the EK is valid and creates a challenge by encrypting an importable *restricted* KEYEDHASH *sign* key to the validated EK. The Verifier provides the challenge to the Host.

Note:

The restricted HMAC key is ephemeral for the protocol. It needs to be randomly generated by the Verifier and used only within this specific protocol instance.

Note:

The Verifier needs to remember the restricted HMAC key for a later step. It might do this by providing an encrypted opaque context structure to the Host to be provided later, or through some other means.

3. The Host imports the restricted HMAC key (`TPM2_Import()`) under the EK to produce a loadable restricted HMAC key blob, and loads (`TPM2_Load()`) that blob.
4. The Host generates (`TPM2_Create()`) and loads (`TPM2_Load()`) the AK for which it will obtain a credential.

   Note:

   Figure 3 specifically depicts a `TPM2_Create()`-based flow, but `TPM2_CreatePrimary()` could be used instead.

5. The Host certifies the AK using the restricted HMAC key (`TPM2_Certify()` or `TPM2_CertifyCreation()`).

   Note:

   This same flow works for both the `Certify` and `CertifyCreation` use-cases. `CertifyCreation` allows the Verifier to have provided an additional nonce (*qualifyingData*) before the AK was created in step 1, which can be used to prove that the AK was generated after the nonce was generated.

6. The Host provides the HMAC-signed attestation structure to the Verifier.
7. The Verifier checks the HMAC-signed attestation structure:

   - checking the *firmwareVersion* number from the attestation statement (outer `TPMS_ATTEST`) against expected values, or recording it for later reference
   - checking the *name* from the attestation statement (inner `TPMS_CERTIFY_INFO` or `TPMS_CREATION_INFO`) against the expected Name hash for the AK public key

   The Verifier can choose to enforce a policy on which firmware versions are acceptable, prior to issuing the AK credential. The Verifier can also embed the firmware version within the AK credential.

8. The Verifier issues the AK credential and provides it to the Host.

   The Verifier optionally stores the information about the attested firmware version in its database associated with the EK credential for later use or audit.

3.1 Privacy Concerns

Both the existing `ActivateCredential`-based AK protocol and this document's `Import`/`Certify`-based AK protocol support a privacy-preserving mode. In the privacy-preserving mode of both the `ActivateCredential`-based AK protocol and the `Import`/`Certify`-based AK protocol, the Host sends its own TPM's EK credential along with a number of other authentic EK credentials, and the Verifier responds by encrypting the same data to all the EKs. In the `ActivateCredential`-based AK protocol, the same "challenge" data (e.g., nonce) is encrypted to each EK. In this document's `Import`/`Certify`-based AK protocol, the same restricted HMAC key is encrypted to each EK. For more information on the privacy-preserving features of the EK, see the TPM Library [2] Part 1: "Credential Protection."

4 Variant: Firmware Version retrieval independent of AK deployment

It is not necessary to tie the firmware verification to the deployment of an AK. In several scenarios, it is easier to verify only the TPM's firmware associated with the EK and e.g. store this information in a database. One such scenario is the validation if a firmware update for the TPM was actually applied to a TPM or whether this process was interrupted or circumvented. The flow is depicted in Figure 4.

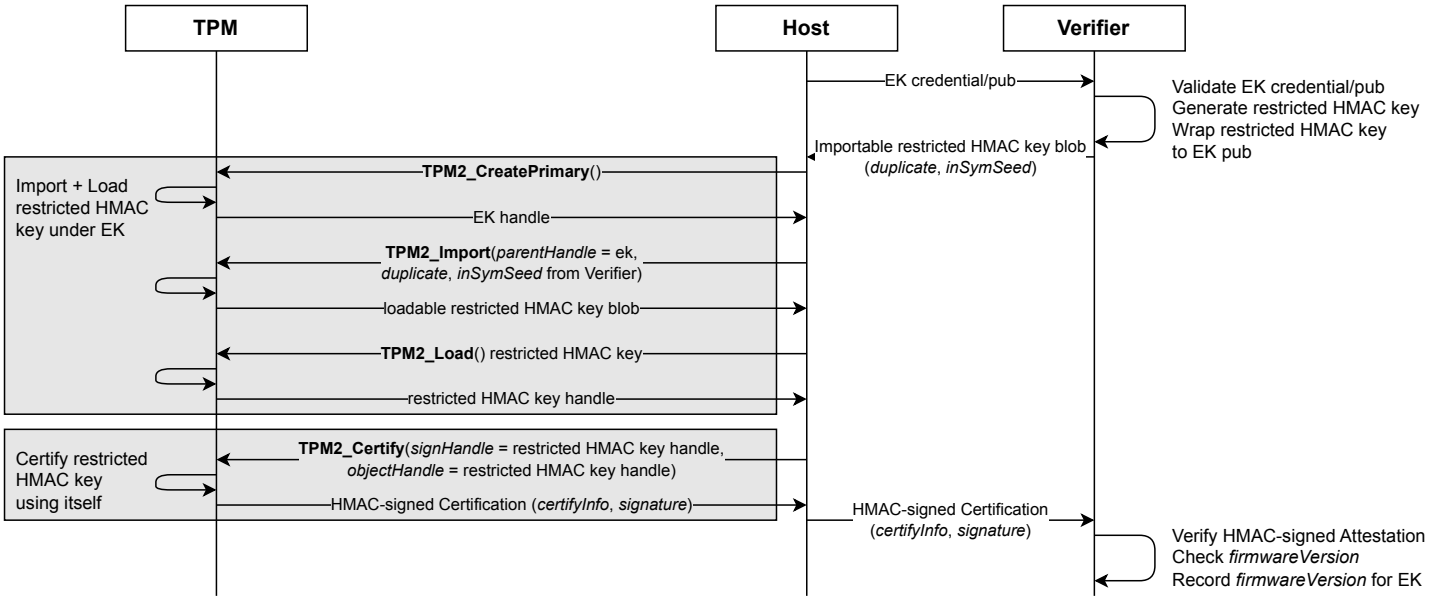

Figure 4: Protocol Variant Overview

1. The Host provides the EK public key or credential(s).
2. The Verifier checks that the EK is valid and creates a challenge by encrypting an importable *restricted* KEYEDHASH *sign* key to the validated EK. The Verifier provides the challenge to the Host.

> ***Note:***
>
> The restricted HMAC key needs to be randomly generated by the Verifier and used only within this specific protocol instance.

> ***Note:***
>
> The Verifier needs to remember the restricted HMAC key for a later step. It might do this by providing an encrypted opaque context structure to the Host to be provided later, or through some other means.

3. The Host imports the restricted HMAC key (`TPM2_Import()`) under the EK to produce a loadable restricted HMAC key blob, and loads (`TPM2_Load()`) that blob.
4. The Host certifies the restricted HMAC key with itself using `TPM2_Certify()`.

> ***Note:***
>
> This same flow can be adapted for any attestation command supported by the TPM, because every attestation statement produced by the TPM contains *firmwareVersion*.

5. The Host provides the HMAC-signed attestation structure to the Verifier.

6. The Verifier checks the HMAC-signed attestation structure and the *firmwareVersion* number from the attestation statement (outer `TPMS_ATTEST`). The *name* from the attestation statement (inner `TPMS_CERTIFY_INFO` or `TPMS_CREATION_INFO`) can be ignored.

   The Verifier stores the information about the actual current firmware in its database associated with the EK credential for later use or audit.

References

[1] “TCG EK Credential Profile for TPM Family 2.0.” Trusted Computing Group, Dec. 2024. Available: https://trustedcomputinggroup.org/resource/http-trustedcomputinggroup-org-wp-content-uploads-tcg-ek-credential-profile-v-2-5-r2_published-pdf/

[2] “Trusted Platform Module Library Specification, Family 2.0, Level 00, Revision 01.83.” Trusted Computing Group, Mar. 2024. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/

[3] “TCG Platform Certificate Profile.” Trusted Computing Group, Jul. 2024. Available: https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/

