TCG Infrastructure WG
TPM Keys for Platform Identity for TPM 1.2

Specification Version 1.0
Revision 3
21 August 2015
Published

Contact: admin@trustedcomputinggroup.org

TCG

TCG Published

Table of Contents

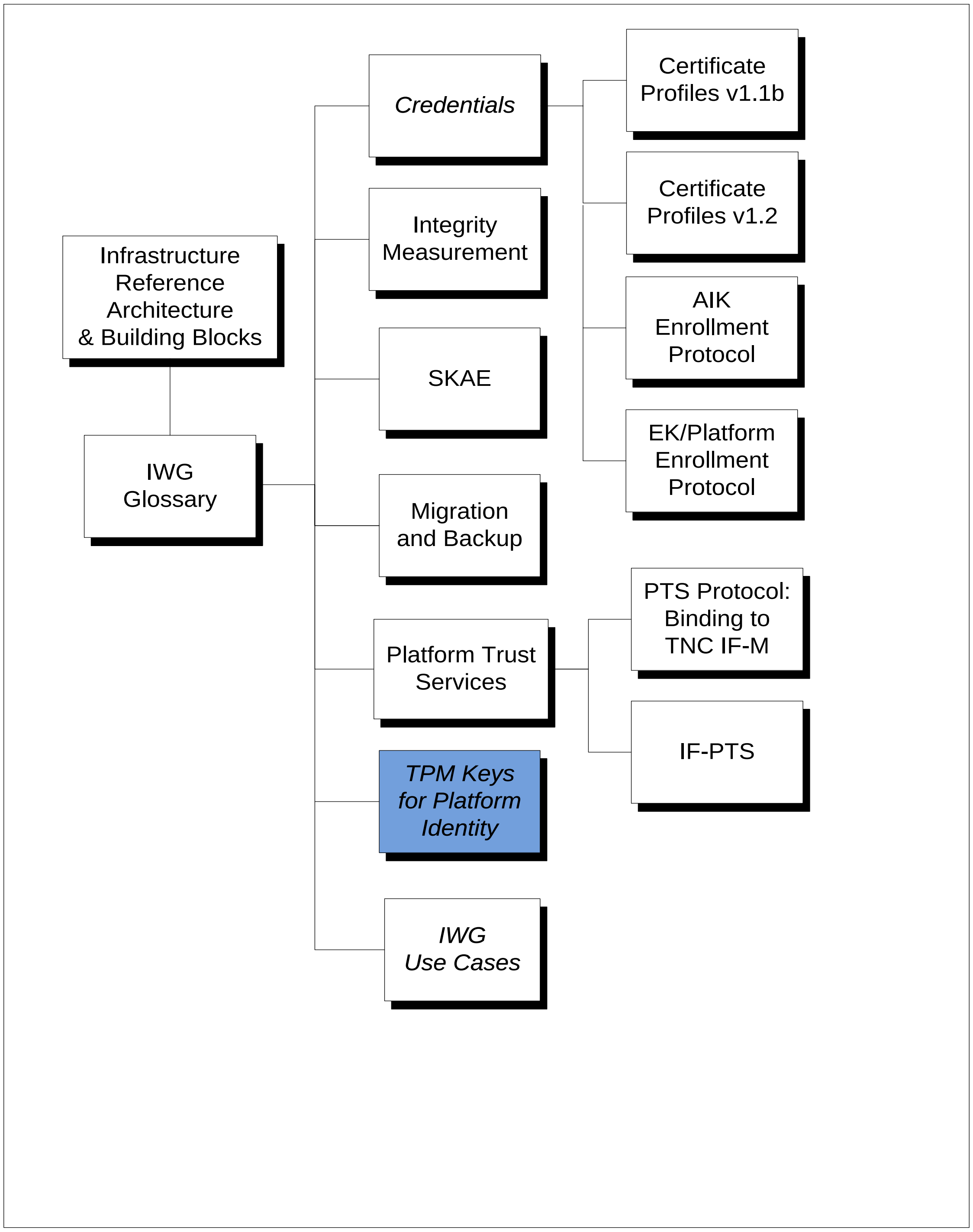

Infrastructure Reference Architecture & Building Blocks
IWG Glossary
Credentials
Integrity Measurement
SKAE
Migration and Backup
Platform Trust Services
TPM Keys for Platform Identity
IWG Use Cases
Certificate Profiles v1.1b
Certificate Profiles v1.2
AIK Enrollment Protocol
EK/Platform Enrollment Protocol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

Figure 1. IWG Documentation Roadmap

Acknowledgement

The TCG wishes to thank those who contributed to this specification. This document builds on considerable work done in the various working groups in the TCG.

Special thanks to the members of the IWG and others contributing to this document.

Name	**Member Company**
Ronald Aigner	Microsoft
Bob Bell	Cisco
Rene Bourquin	General Dynamics
Mike Boyle	United States Government
Alvaro Cardenas	Fujitsu
Dave Challener	Johns Hopkins
Ga-Wai Chin	Infineon
Olivier Collart	STMicroelectronics
Michael Cooper	United States Government
Carlin Covey	Freescale
Emily Doll	United States Government
Michael Donovan	Hewlett-Packard
Paul England	Microsoft
Junfeng Fan	Nationz
Jessica Fitzgerald-McKay	United States Government
Ken Goldman	IBM
Anne-Roe Gratadour	Thales
Markus Gueller	Infineon
Steve Hanna	Infineon
Frederic Hannoyer	STMicroelectronics
Eric Hibbard	Hitachi
Wyllys Ingersoll	Oracle
Bill Jacobs	Cisco
Masayaki Kanda	IPA
Sotora Kanno	IPA
Greg Kazmierzcak	Wave

Scott Kelly	Hyperthought
Tom Laffey	Hewlett-Packard
Carolin Latze (Editor, Co-Chair)	Swisscom
Sunjae Lee	Samsung Electronics
Rafael Montalvo	Cisco
Gilles Peskine	Gemalto SA
Stanley Potter	United States Government
Max Pritikin (Co-Chair)	Cisco
Padma Prishnaswamy	Battelle Memorial
Graeme Proudler	Hewlett-Packard
Emily Ratliff	AMD
James Raune	Broadcom
Lisa Raykowski	DMI
Allen Roginsky	NIST
Anders Rundgren	PrimeKey Solutions
Paul Sangster	Symantec
Johann Schoetz	Infineon
Ariel Segall	The MITRE Corporation
Gloria Serrao (Editor)	United States Government
Ari Singer	DMI
Todd Slack	Atmel
Ned Smith	Intel
Stephen Squires	Battelle Memorial Institute
Andreas Steffen	HSR
Gregg Tally	Johns Hopkins
Sander Temme	Thales
Josephine Vlastaris	Wave Systems
Monty Wiseman	Intel
David Woolf	UNH Interoperability Lab
Jeremy Wyant	General Dynamics C4 Systems

Special thanks to Scott Kelly, who reviewed and improved all the text around EK, AIK, and certificates in general, to Tim Meggison, Jeremy Wyant, Leslie Andresen, and Joyce Fai for all the discussions around device identity provisioning, Kelly Burgin to improve the sections about IKE, Tom Yu, Greg Hudson, and Thomas Hardjono who reviewed and improved the section about PKINIT, Max Pritikin who made sure we use the normative language as we should, and Ariel Segall, Jessica Fitzgerald-McKay, Tom Laffey, Steve Hanna, Greg Kazmirzcak, Andreas Steffen for countless reviews and improvements.

Carolin Latze

Specification Editor

Introduction

1.1 Scope and Audience

The Infrastructure Working Group within the TCG is defining an open set of security technologies providing a common infrastructure across the different TCG solution sets. This specification addresses ways to incorporate TPM created keys into solutions for device identities. It addresses how the resulting device identities interface with and are represented within an existing public key infrastructure. This specification uses the IEEE Standard for Local and Metropolitan Area Networks, Secure Device Identity (802.1AR) [4] device identity module definition and formatting.

Secure device identities that are not easily spoofed or copied from memory are of paramount importance to a secure network and can be used in conjunction with network access control solutions to protect the network against devices that contain malware and to prevent easy access to devices that allow network access.

This specification will address which TPM keys are to be used, how they are reflected in an X.509 certificate and how consuming parties will be able to recognize and use device identities to determine authentication to a network within existing authentication protocols. The methods of using TPM based keys presented in this specification overcome the Attestation Identity Key (AIK) usage limitations of TPM 1.2 and for that reason do not use the AIK key itself, but instead a key that has been "certified" (or signed) by the AIK by using the TPM_CertifyKey command specified in [44]. This AIK "certification" means the TPM can assert that the private key of the device identity is held securely in the TPM (for details see section 3.1). The use of non-certified TPM signing keys to assert TPM residency is out of scope of this specification.

Architects, designers, developers, and technologists interested in the development, deployment, and interoperation of trusted systems will find this document necessary when making use of TPM based keys within certificate structures and across network enterprises as identities for devices, which can include non-person entities [38].

Before reading this document any further, the reader should review and understand the IWG architecture as described in [1] and [2]. The reader should also be familiar with the IEEE 802.1AR standard as published in [4].

This specification applies only to TPM 1.2 and below.

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.

2 Background

2.1 Role of TPM-based Platform Identities

This document describes standard ways to bind TPM-based key material within X.509 certificate structures so that communities can share and utilize device identities that are based on TPM-resident keys. This specification describes TPM-based cryptographic primitives and certificate structures so that device identities can be based on the TPM hardware module and be interoperable across user communities. The TPM provides enhanced security for identifiers because of its ability to protect the private key in hardware. In this way a cryptographic identity is bound to a device in which a particular TPM exists.

TPM-based platform identities created, provisioned and used as directed in this specification can:

- Provide for centralized asset management
- Provide hardware-based identities that cannot be easily spoofed in the way that IP and MAC addresses can.
- Provide a standards-based hardware identity
- Provide a basis for device-to-device authentication for example:
  - Act as the first step in securing an endpoint. The identity will be used for authentication, which can then optionally be followed by TPM based integrity measurements. Provide strong identities to be used in standard authentication protocols such as Transport Layer Security (TLS), Internet Key Exchange (IKE) within IPsec, and PKINIT (Public Key Cryptography for Initial Authentication for Kerberos), and in Simple Authentication and Security Layer (SASL) mechanisms.
  - Provide identity support for delivery of software (license keys, algorithms and security software)

2.2 IEEE 802.1AR

The IEEE 802.1 working group defined the IEEE 802.1AR Standard for Local and Metropolitan Area Networks Secure Device Identity and it was published on 22 December 2009 [4]. This standard defines a per-device unique identity installed at manufacturing time and used subsequently in device-to-device authentication exchanges. This standards-based device identity can also be coupled in multiple ways with user identification as the use cases, below, point out.

The 802.1AR standard defines a secure device identifier (DevID) as “a cryptographic identity that is bound to a device and used to assert the device’s identity”. It further specifies:

- the DevID is an X.509 credential
- globally unique per-device identifiers and the management and cryptographic

binding of a device to its identifiers

- the relationship between an initially installed identity and subsequent locally significant identities, and
- interfaces and methods for use of DevIDs with existing and new provisioning and authentication protocols

The initially installed identity is defined as an IDevID ("I" for initial) and subsequently locally defined identities are LDevIDs ("L" for local). The IDevID is installed at manufacturing time. Since this "TPM Keys for Platform Identity" specification allows for an on premise creation of the device identity, it will use the term DevID. Whether or not this DevID is an IDevID or an LDevID is of no importance to the TPM. That depends on the use case. An IDevID will be created at manufacturing time and allows proofing that this device has been manufactured by a certain manufacturer. An LDevID is created on premise and allows proofing that this device is owned by a certain enterprise (or private person).

There is another dimension to the IDevID/ LDevID discussion with TPMs prior to and including version 1.2. Creating a DevID requires taking ownership of the TPM. In order to create an IDevID, the platform manufacturer needs to take ownership of the TPM. There is a command to change the ownership of a TPM, however that makes all previously created keys unusable. When considering what rights the device manufacturer is going to allow to pass out of his control after creating an IDevID consideration must be given to make sure that the IDevID cannot be erased from the TPM. This means that the end user should not be able to change the SRK or evict the IDevID key. If the user needs to use other normally owner authorized commands, delegation commands can be used.

For the reasons just explained, the remainder of this specification will only talk about DevIDs.

The 802.1AR specification DevID module contains a service interface, storage holding a DevID secret and credential, secure hashing functions, a random number generator (RNG) and asymmetric cryptography functions. These functions exist in the TPM and calls by middleware (such as the TCG software stack (TSS) or the TPM API) can be used to meet interface requirements outlined in 802.1AR.

The IEEE Standard 802.1AR-2009 [4] can be used together with TPM-based keys and certificates. The TPM acts partly as the Secure Device Identifier Module (DevID Module) which the standard defines as "a logical security component that will secure, store and operate on one or more DevID Secret(s) [(private key)] and associated DevID credentials". While the TPM can store the keys itself, the certificate chain will be stored outside the TPM. This document addresses usage of the TPM after provisioning has occurred and leverages a TCG private X.509 certificate extension that proves TPM residency of the keys, this extension is the Subject Key Attestation Evidence (SKAE) extension.

This "TPM Keys for Platform Identity" specification defines how TPM keys and the resulting certificate be used as components of the DevID. The actual identity will be reflected in the X.509 certificate.

2.3 TPM-based DevID and 802.1AR Requirements

Implementations that are conformant to this document provide an implementation of the IEEE 802.1AR specified functionality. The following tables provide a populated "Protocol Implementation Compliance Statement" (PICS) template for IEEE 802.1AR, as defined in Annex A of that document. Implementers of this specification are to provide a complete IEEE 802.1AR PICS to consumers of their products that also includes the IEEE 802.1AR "Implementation identification" table.

Protocol summary:

Identification of protocol specification	**IEEE Standard 802.1AR-2009, IEEE Standard for Local and Metropolitan Area Networks—Secure Device Identity**	**Additional Information**
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	None.	
Are there exception items? (See A.3.3: The answer Yes means that the implementation does not conform to IEEE Standard 802.AR-2009.)	No [ ] Yes [ ]	Section 3.1 explains that the TPM provider can choose to include the IDevID or not. If the TPM provider chooses not to include the IDevID then an exception is indicated. See IEEE 802.1AR section A.3.3.

Major capabilities and options:

Item	**Feature**	**Status**	**Support**	**Additional Information**
IDEV	Contain an Initial Device Identifier (IDevID)	M	Yes [Section 3.1]	Section 3.1 explains that the TPM provider can choose to include the IDevID or not. If the TPM provider chooses not to include the IDevID then an exception is indicated. See IEEE 802.1AR section A.3.3.
DEVIDF	Include specified fields in all DevIDs	M	Yes [Section 3.2]	
CHAIN	Contain an IDevID	M	Yes [TSS Specification	The IDevID certificate chain will be stored on

	certificate chain		[46]]	persistent storage
KEYRSA	Support RSA asymmetric cryptography	O.1	Yes	Specified in TPM 1.2 specification [13]
KEYECDSA	Support ECDSA asymmetric cryptography	O.1	No	Not supported in TPM 1.2 and below.
STOR	Meet the minimum requirements for secure storage	M	Yes	Secure Storage via SRK [13]
DEVSI	Meet the minimum requirement for the service interface	M	Yes	See table below for more details.
EDEVSI	Provide for some or all of the LDevID service interface functionalities	O	Yes	See table below for more details
CHAINF	Include specified additional fields for DevID certificate chains	M	Yes	This is a requirement to be addressed by the TPM provider.
SERIALNUM	Does the subject name field include the unique platform serial number?	O	Yes	The serial number defines a platform. It is left to the manufacturer of a platform to set this field. 3.2.1
HWMODNAME	Does the subjectAltName field include the device hardwareModuleName	O	Yes	See section 3.2.1 for more details.
SNMPv3	Support access to the MIB using SNMP v3	O	No	This details a Mandatory feature that is not provided by the TPM itself. The vendor of the product that uses the TPM provides these functionalities according to the product requirements.

<u>DevID Service Interface:</u>

The DevID Service Interface is implemented by a "platform module" written in software in concert with the underlying TPM module.

The table below enumerates the mandatory 802.1AR requirements for a Service Interface and matches these requirements to TPM 1.2 commands when applicable. When there is not a direct correlation, the 802.1AR requirement is described so that an implementer is aware of requirements to be implemented by another component of the solution (i.e. the BIOS, or the operating system.)

Item	**Feature**	**Status**	**TPM Support**	**Reference**
SI-1	Initialization	M	Yes – platform module	The TPM does not perform this requirement. The requirement per 802.1AR is "to cause the IDevID module to prepare itself for use. This operation is to be used by the OS or firmware to properly initialize the IDevID. This initialization function will return "FALSE" if its internal checks fail, if it detects a compromise of protected data or if ownership has been successfully assessed. Inputs: None Output: A binary value indicating availability (TRUE) or unavailability (FALSE) of the other IDevID service interface operations. The TPM does have a self test command. In order to ensure that the initialization is done, one has to check the results of that self test command using TPM_GetTestResult.
SI-2	Enumeration of the DevID public keys	M	Yes	The TPM would use the TPM_GetPubKey command when the owner of a key wants to obtain a public key value from a loaded key. This requires authorization from the key owner. Furthermore, the TSPI_Context_GetRegisteredKeysByUUID function could be used on the TSS level. 802.1AR requires an administrator to see a table of public keys and an integer key index value.
SI-3	Enumeration of the DevID credentials	M	Yes – platform module	802.1AR requires that the administrator be able to determine all DevID credentials installed on the DevID module. It requires them to be indexed and returned as the "credentialIndex." A negative value

				indicates an internal error. The IDevID credential is always at index 0. Output: A table containing, for each credential, the credential's credentialIndex, the associated key's keyIndex, a value indicating if the credential is enabled and the credential itself. For the TPM this function will be provided in the platform software.
SI-4	Enumeration of the DevID credential chain	M	Yes – platform module	802.1AR requires this operation to allow the administrator to enumerate a credential chain associated with a specified DevID credential. This could include the root certificate. Output: A table containing each credential within the credential chain. For the TPM this function is provided in the platform software.
SI-5	Signing	M	Yes – Platform module	The TPM_Sign command signs data and returns the resulting digital signature. In order to support this, the key is generated as a "PKCS1_V15_DER" key; which allows the equivalent of PKCS1DIGEST_INFO_OPAQUE. Using this the platform module can implement PKCS1HASH_SHA256. ECDSA is NOT supported in TPM 1.2 Per 802.1AR: Input: keyIndex, CurrentEncoding, dataLength, dataOctets Example: Enum currentEncoding_t {PKCS1HASH_SHA256, PKCS1DIGESTINFO_OPAQUE, ECDSADIGESTINFO_OPAQUE} Output: A value indicating success or failure of the operation, the signature.
SI-6	Enable/Disable DevID credential	M	No – Yes – via platform module.	Per 802.1AR, this command is used by the administrator to control the availability of DevID credentials. A specific DevID credential can be disabled. The operator allows the device administrator to prevent use of a

				particular credential without disabling the associated key, which can be used by another credential. Re-enabling a disabled DevID credential makes it fully available. Input: The credentialIndex of the DevID credential to be enabled or disabled. The desired state (enable or disable). Output: A value indicating success or failure of the operation. This function will be provided by the platform software.
SI-7	Enable/Disable DevID key	M	Yes In order to access a key in the TPM, the key may require an authenticated session. This mechanism is used to enable and disable keys. As long as the authenticated session exists, the key is enabled and can be used. Once it is closed, the key is disabled.	Per 802.1AR, this command is used by the administrator to control the availability of DevID keys. A specific DevID key can be disabled. Re-enabling a disabled DevID key makes it fully available. Input: The index of the DevID key to be enabled or disabled. The desired state (enable or disable). Output: A value indicating success or failure of the operation.

2.4 Rationale for TPM Keys Used and Their Relation to IEEE 802.1AR

Because the Endorsement Key (EK) has a long lifetime and is persistent, using it as the DevID initially appears logical. However, this is not practical because of the constraints on the TPM 1.2 EK, which require that it only be used for encryption. The Attestation Identity Key (AIK) is designated as TPM_KEY_IDENTITY key, which is an RSA key to which the TPM assigns unique properties. The AIK has constraints in that it can only sign structures that originate in the TPM itself. This specification will refer to the AIK as an “identity” key, which differs from the DevID authentication key. The DevID is created from a TPM generated, AIK certified, non-migrateable signing or general purpose key.

A lot of the authentication protocols in use today require encryption to be used. A general purpose key can do signing (needed to sign a CSR) and encryption operations and is therefore the right choice if an implementer is unsure about the authentication protocol. If it is certain that the authentication protocol to be used supports signing operations, it is also safe (and better) to choose a signing key.

The TPM public/private key pair to be used as the DevID are keys that have been certified by the AIK. It can be a general purpose or a signing only key and thus are TPM_KEY_LEGACY (“general purpose”) or TPM_KEY_SIGNING (“signing”) per the TPM 1.2 Specification. Be aware that there are some TPMs in the field that disallow for legacy keys in FIPS 140-2 mode. If in doubt, check the technical specification of your TPM.

This specification will use the terms DevID “authentication key” to differentiate it from the AIK identity key described above. The DevID authentication key is used within the authentication protocols enumerated herein.

Although the key that is used within the DevID is important, the DevID itself is the X.509 certificate associating that key (referred to as authentication key in this document) to the device. The certificate contains Subject Name field and serial number attribute (per 802.1AR) if it is created by an OEM. If it is created by an Enterprise, the Subject Name field is optional. In both cases, however, the SubjectAltName MUST be included and the TPM is reflected in the HardwareModuleName extension as described in detail in Section 3.2.

The DevID representation within the certificate in the context of this specification contains a TCG OID that represents TPM Version 1.2 in the hwType field together with a unique value in hwSerialNum (often TPM serial number) as assigned by the manufacturer.

2.5 Supported Use Cases

The following use cases demonstrate how the TPM-based keys and certificates are used for device authentication exchanges that utilize X.509 certificates.

In the following use cases, a TPM-based non-migratory AIK certified RSA authentication key will be used together with the associated X.509 certificate. The term “certified key” refers to the output of the TPM_CertifyKey function as specified in [44]. A certificate authority signature and certificate policy binds the TPM authentication key acting as the DevID to the TPM for most of the use cases.

2.5.1 Single Authentication Certificate: Client-side Device Identity

Use an X.509 certificate of the format described in IEEE 802.1AR (DevID certificate) as the client-side certificate for authentication within the secure session establishment. The following protocols are examples of protocols that support X.509 certificate-based authentication: TLS, IKE within IPsec, PKINIT and SASL.

This use case uses only one client-side authentication certificate that is the DevID certificate.

In this use case the use of Subject Key Attestation Evidence (SKAE Extensions) is optional for verifying parties. If the CA includes a CP extension in the certificate indicating that the SKAE was verified as a condition of issuance, then inclusion of the SKAE extension in the DevID certificate is optional. SKAE extensions are used to link an Attestation Identity Key (AIK) to a certified key pair because the AIK is used to sign information contained in the SKAE. This use case allows for a CA signature and certificate policy to link the initial certificate request signed by the DevID to the SKAE and prior AIK signature within the SKAE to the TPM and not require SKAE parsing by validating parties. SKAE extensions can also be used if more assurance is desired.

2.5.2 Single Authentication Certificate: Server or Infrastructure Component

The TPM-based 802.1AR certificate and private authentication key as described above can be used as the device identity for servers, other network components (firewalls, switches, etc.) and other devices that may need an identity but that are not associated with a user (printers, etc.).

In this case, the server or infrastructure component will use its device identity certificate to authenticate within TLS, IKE within IPsec, PKINIT and SASL mechanisms. It may authenticate to a remote authentication server or the exchange could be part of a peer-to-peer authentication scheme.

In this use case the use of Subject Key Attestation Evidence (SKAE Extensions) is optional for verifying parties. If the CA includes a CP extension in the certificate indicating that the SKAE was verified as a condition of issuance, then inclusion of the SKAE extension in the DevID certificate is optional. SKAE extensions are used to link an Attestation Identity Key (AIK) to a certified key pair because the AIK is used to sign information contained in the SKAE. This use case allows for a CA signature and certificate policy to link the initial certificate request signed by the DevID to the SKAE and prior AIK signature within the SKAE to the TPM and not require SKAE parsing by validating parties. SKAE extensions can also be used if more assurance is desired.

2.5.3 Single Authentication Certificate: User/Device Authentication in Public Wireless Networks (IEEE 802.1X/ EAP-TLS) Using TLS Extensions

Nowadays, user authentication in ISP operated IEEE 802.1X-based public wireless networks (PWLAN) is the same as device authentication. People tend to have their personal device represent their identity, i.e. the devices are not shared. Such a device is then bound to one user by contract. Furthermore, authentication in PWLANs need provide only “enough security” to allow for accounting and incident tracking. That requires some kind of hardware-protected identity to be used. The TPM is built into a lot of new devices and it is likely that they are available in other mobile devices in the future too. TPM-protected keys and the resulting certificates are useful for device authentication as already stated in [4]. Having a valid TPM-based credential, a user can authenticate her device at a PWLAN hotspot without the need for further certificate/ identity requests.

In order words: For this use case, the relying party needs a proof of possession (PoP) of a key, but can ignore the RFC 5280 [10] validation of the certificate. This can be achieved using an alternative (non-X.509) trust chain: The credential used for authentication will be a TPM-based key certified by an AIK, which is stored as self-signed X.509 certificate. The binding between the AIK and the certified authentication key will be evidenced by an AIK certificate sent within a TLS extension. It has to be mentioned that although this use case specifies one user per device, it applies as well for multiple users on the same device. Every user would have a separate AIK and therefore a separate AIK certified authentication key and certificate.

Using TPM-based certified keys during EAP-TLS authentication within a PWLAN environment will reduce the number of CA requests to a network provider. Only an AIK certificate is needed, not a CA signed certificate for the certified authentication key. This will reduce the operational burden and cost of multiple CA requests for PWLAN authentication.

This use case requires that the SKAE extension be processed by the validating parties. In addition, the AIK certificate is used to validate the AIK signature over information in the SKAE.

As a final comment, although device authentication is considered sufficient for this use case, an additional user authentication based on this device authentication is still allowed. However, user authentication following device authentication as described in this use case is not covered within this specification. Any standard user authentication will do.

2.5.4 Device and User Identity Certificates within two Separate Network Security Protocols

This use case supports both a device and user-based authentication at the same time, but within two unique network security protocols. The binding of the TPM-based device identity and the user identity is performed within a Metadata Access Point (MAP) server per the TCG IF-MAP specifications [41] or via another secure back-end database. It does not address two authentications within one protocol session (IPsec/TLS) since that is handled in section 2.5.7.

A TPM-based device identity and a user TPM-based identity certificate are created within two separate security protocols on a network (for example a layer 2 VPN (802.1X and RADIUS) and a layer 3 IPsec exchange). X.509 certificates are used for both identities. The authentication endpoints can be different (e.g. a gateway for device authentication and an authentication server for user authentication) or they can be the same.

The device identity will be bound to the user identity within a secure database such as IF-MAP.

If desired, access decisions can be incremental. Initial decisions about the level of network access to be granted can be made based on the device identity and the subsequent user authentication can determine the correct access levels to be granted to information and data stores. Or, access decisions can be made on the presence or absence of both certificates in the secure database.

In this use case the use of Subject Key Attestation Evidence (SKAE Extensions) is optional for verifying parties. If the CA includes a CP extension in the certificate indicating that the SKAE was verified as a condition of issuance, then inclusion of the SKAE extension in the DevID certificate is optional. SKAE extensions are used to link an Attestation Identity Key (AIK) to a certified key pair because the AIK is used to sign information contained in the SKAE. This use case allows for a CA signature and certificate policy to link the initial certificate request signed by the DevID to the SKAE and prior AIK signature within the SKAE to the TPM and not require SKAE parsing by validating parties. SKAE extensions can also be used if more assurance is desired.

2.5.5 Device and User Identity Certificates within Tunneled EAP

In this use case, both a user identity certificate and a device identity (an 802.1AR DevID) are used to authenticate the endpoint within a tunneled EAP exchange e.g. to run a TNC protocol.

Chained EAP methods support this use case. Examples are EAP-FAST or TTLS, both of which are draft RFCs, with a standard method for tunneled EAP, which has been determined by the IETF EAP Methods Update (EMU) working group in [31]. Using tunneled EAP, two certificates are provided for authentication. The first one is needed to authenticate the device, and a subsequent certificate to authenticate the user.

In this use case the use of Subject Key Attestation Evidence (SKAE Extensions) is optional for verifying parities. If the CA includes a CP extension in the certificate indicating that the SKAE was verified as a condition of issuance, then inclusion of the SKAE extension in the DevID certificate is optional. SKAE extensions are used to link an Attestation Identity Key (AIK) to a certified key pair because the AIK is used to sign information contained in the SKAE. This use case allows for a CA signature and certificate policy to link the initial certificate request signed by the DevID to the SKAE and prior AIK signature within the SKAE to the TPM and not require SKAE parsing by validating parties. SKAE extensions can also be used if more assurance is desired.

2.5.6 Device and User Identity Certificates within TLS using Secure Renegotiation

TLS and TLS-based protocols such as EAP-TLS can use TLS extensions to incorporate TPM-based device identities. This use case supports both a device and a user identity certificate. This use case will be handled by performing multiple TLS certificate handshake sequentially for the same TLS session. Furthermore, the secure renegotiation extension is used to bind them.

In this use case the use of Subject Key Attestation Evidence (SKAE Extensions) is optional for verifying parties. If the CA verified the SKAE extension during the initial CSR processing and stated that in the CP extension by including an appropriate OID, it is not mandatory to include SKAE into the DevID certificate. SKAE extensions are used to link an Attestation Identity Key (AIK) to a certified key pair because the AIK is used to sign information contained in the SKAE. This use case allows for a CA signature and certificate policy to link the initial certificate request signed by the DevID to the SKAE and prior AIK signature within the SKAE to the TPM and not require SKAE parsing by validating parties. SKAE extensions can also be used if more assurance is desired.

2.5.7 Device and User Identity Certificates within one authentication protocol

Implementations may support client/entity authentication within one authentication e.g. through the use of SASL mechanisms [18] [19] used in conjunction with any standard authentication protocol that supports SASL.

Some real world examples of this use case are the so called bring-your-own-device (BYOD) initiatives. In BYOD, a company allows employees to bring their own hardware to access the company's infrastructure. This is risky since they still want to make sure that only this employee can access the infrastructure and not any other user that could exist on that device. Therefore, the company could issue a device certificate for this device as well as a user certificate for the employee in order to make sure that only this employee can access the network with his device.

In this use case the use of Subject Key Attestation Evidence (SKAE Extensions) is optional for verifying parties. If the CA includes a CP extension in the certificate indicating that the SKAE was verified as a condition of issuance, then inclusion of the SKAE extension in the DevID certificate is optional. SKAE extensions are used to link an Attestation Identity Key (AIK) to a certified key pair because the AIK is used to sign information contained in the SKAE. This use case allows for a CA signature and certificate policy to link the initial certificate request signed by the DevID to the SKAE and prior AIK signature within the SKAE to the TPM and not require SKAE parsing by validating parties. SKAE extensions can also be used if more assurance is desired.

2.6 Non-supported Use Cases

Authentication methods that do not use X.509 certificates are not supported by this specification.

Those protocols not explicitly named in this specification are also not supported but it is likely that an implementer could use the information contained here to successfully use TPM-based identities in other X.509 certificate-based authentication protocols.

2.7 Requirements

This section lists the requirements expected for a Device Identity in order to successfully interoperate with the TNC architecture. These are stated as general requirements with specific requirements enumerated in the following sections.

2.7.1 Use Standard Authentication Protocols

This document defines a TPM-based device identity that is able to be used by standards-based authentication protocols that are currently implemented and in use.

The TPM-based identity can be used as an additional authentication component within protocols that support two X.509 certificates (for user and machine identification) such as shown in Chapter 7. The protocols described in this specification will work with existing libraries (i.e., TLS libraries). If a protocol does not recognize the TPM-based identity, the authentication process will either fail with a well-known error message or fallback to another (supported) authentication method. For TLS based protocols utilizing extensions, at least TLS version 1.0 is required.

2.7.2 Utilize X.509 Certificate Standards

The information for a TPM-based device identity is reflected within a standard X.509 certificate and managed by standard X.509 certificate management protocols [9].

2.7.3 Utilize the TNC Architecture

The TPM-based identity is used within the TNC architecture as mentioned in use case 2.5.5. Although other protocols are covered within this specification, the TPM-based device identity is compatible with the IF-T transport layer protocols of the Trusted Network Connect (TNC) Working Group of the Trusted Computing Group (TCG) [14][15] and the IETF Network Endpoint Assessment (NEA) Working Group "PT-TLS: A TCP-based Posture Transport (PT) Protocol" [28] and "PT-EAP: Posture Transport (PT) Protocol for EAP Tunnel Methods" [32].

2.8 Non-Requirements

None.

2.9 Assumptions

Here are the assumptions that the TPM Keys for Platform Identities specification makes about other components in the architecture.

- The X.509 certificate and RSA private authentication key resulting from this specification will be used within existing TLS Version 1.0 (and higher), IKE (within IPsec), PKINIT, and SASL mechanism.
- It is assumed that TPM Ownership has been established. This includes the creation of a storage root key and allows for the creation of other children storage keys that are associated with a 160-bit string which authorizes the use of the key.

2.10 Keywords

IDevID, LDevID, ACA, SKAE, TLS, IEEE 802.1AR, IKE, Tunneled EAP, TPM, PKINIT, CP, CPS, CSR, CA

3 TPM Keys for Platform Identity Protocols

This chapter starts with an introduction into the general PKI requirements in order to use TPM keys for platform identity and ends with the definition of the authentication keys used as well as their certification paths.

The table in section 7.2 shows how the sections in this chapter match with the use cases described in section 2.5.

3.1 Identity Provisioning

The sections below describe the certificates to be created in support of a DevID and the fact that an evidentiary chain links the DevID to a TPM. A provisioning organization will use existing Infrastructure Working Group (IWG) enrollment protocols to create and enroll the TPM specific Endorsement Key (EK) and Certificate and the Attestation Identity Key (AIK) and Certificate.

The AIK is the TPM “identity” key that is used to certify the DevID authentication key. A TPM 1.2 key cannot be created as both an identity key and a general purpose key. Therefore, the AIK acts as the identity key that certifies a general purpose key. When the TPM certifies a key, it provides a signature with a TPM identity key over some information that describes that key. The TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure is the mechanism (information) that allows this certification functionality. This specification uses the AIK certified general purpose key as the DevID (or authentication) key. In order to address both signing and encryption based authentication protocols – the signing or general purpose key will be called “authentication key” for the remainder of this specification

To provide proof of this signature, the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure (which is signed by the AIK) is placed into a Subject Key Attestation Evidence extension, a TCG defined extension within an X.509 certificate. This extension plays an important role in the DevID evidentiary chain and is provided to the CA as a part of the certificate signing request (CSR.)

The Subject Key Attestation Evidence (SKAE) X.509 certificate extension enables the cryptographic binding of TCG-oriented security assertions within a common certificate. Because this extension can be added by a CA to a standards-based certificate, secure communication of TCG properties can occur over widely deployed certificate-based security protocols such as SSL, TLS or IKE (used with IPsec). For example, SKAE includes an assertion that the asymmetric private key (corresponding to an SSL certificate’s public key) was generated and is only accessible to a particular TPM. This information is useful for the relying party to gain confidence in the authenticity of the holder of the key since it is far less likely that the private key was stolen and be used fraudulently by the requestor.

3.1.1 Provisioning Roles and Infrastructure Components

Figure 2 summarizes the roles and components described in the following section.

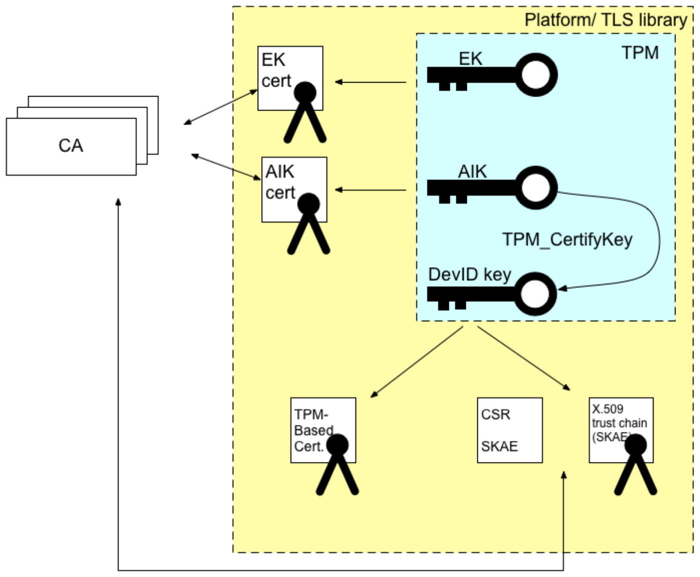


Figure 2 Overview of Roles and Components

3.1.1.1 Certificate Authority

A Certificate Authority (CA) MUST verify TPM residency of the device identity key by parsing the certificate signing request (CSR) and validating TPM AIK signature over the SKAE structure (this structure is named TPM_CERITFY_INFO or TPM_CERTIFY_INFO2.) [39] This structure is contained in the SKAE extension as defined in [39].

The CA SHOULD support a CSR that is sent via a standard protocol. (For example, a PKCS#10 or Certificate Request Message Format (CRMF)). The CA SHOULD also support a standard certificate transport protocol that provides protection from replay attacks and provides for confidentiality and integrity. An example of such a transport protocol is Certificate Management over Cryptographic Message Syntax (CMC) [12].

In addition to validating the signature on this structure, the CA SHOULD also:

- Retrieve the referenced AIK certificate and validate the certificate (signature is good and from a trusted party and check for revocation, etc.)
- Verify the TPM key's public key matches the public key in TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2
    - Validate other TPM properties contained in TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2
        - Key usage: the type of TPM (signature, binding, legacy, Identity, etc.)

- Key flags: migratable, volatile, etc.
- Auth data usage: requires authorization value always, never
- Algorithm params / key size
- PCR status: Is the TPM's key's use gated by PCR state, and if so what state
- Nonce

In order to assert that the CA verified TPM residency for the key used in a DevID certificate, a certificate policy OID unique to the TCG (and for this purpose) MUST be included in the certificate policy extension of the DevID certificate. The CA MUST perform verification of the AIK signature over the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 prior to including the certificate policy TCG OID.

The TCG Policy OID is:

2.23.133.6.1.2

The CA's Certificate Practice statement (or similar policy statement) MUST reflect that the verification of AIK signature is completed prior to issuing device identity certificates.

3.1.1.2 Keys and Certificates Used with CA Trust Chains

The following sections describe the keys that are created as part of the DevID certificate chain.

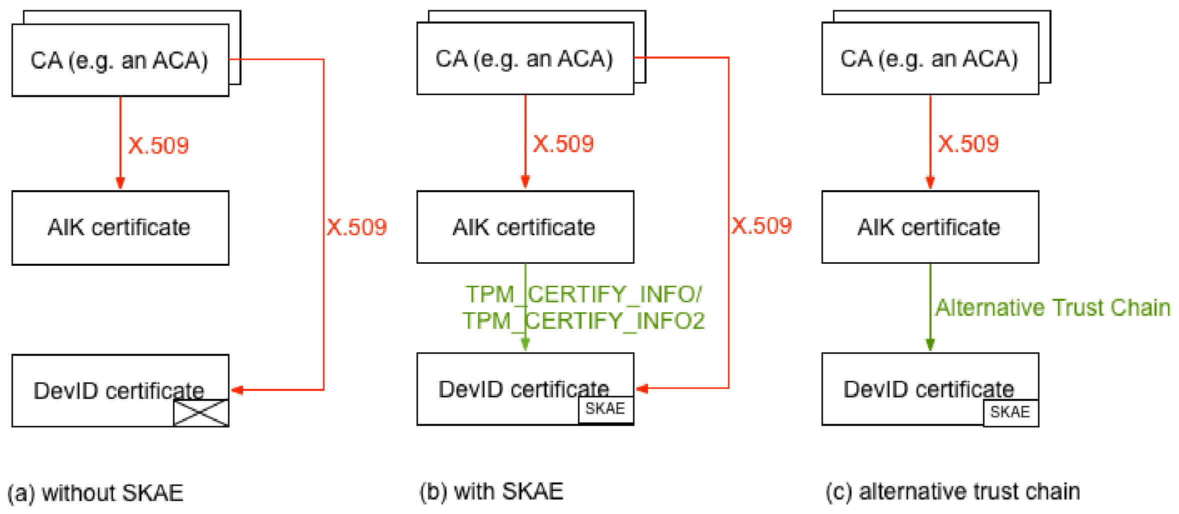


Figure 3 Comparison of Certificate Chains

Figure 3 demonstrates the certificate chains as:

a) The DevID certificate is signed by a CA and this is reflected in standard X.509 certificate fields' subjectKeyIdentifer (In CA certificate) and authoritykeyidentifier (in DevID). Note: Intermediate CAs can also be a part of this valid certificate chain.
b) The DevID is signed by a CA and the standard X.509 certificate chain is available as well as the optional validation that the AIK certificate has signed the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure and that is reflected in the SKAE of the DevID. This case is described in section 3.1.1.2.3.
c) The DevID is technically self-signed and the AIK has signed TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structures of the SKAE. No X.509 certificate standard certificate chaining is exists. This case is described in section 3.1.1.2.4.

3.1.1.2.1 Endorsement Key and Certificate

The EK is an embedded RSA key pair that is a unique identifier of a particular TPM hardware instance. Depending on its use, a user organization could declare the EK and its certificate to be privacy sensitive. The EK cannot be used as a signing key, that is why the Attestation Identity Key (AIK) (another RSA key pair) is created, linked to the EK, and used as an RSA key for signing TPM created structures.

If the EK has not been enrolled by the manufacturer, the Endorsement Key and certificate MUST be enrolled as directed by the IWG EK enrollment specification [47]. The EK is required to create the TPM unique SKAE extension.

3.1.1.2.2 Attestation Identity Key and Certificate

The Attestation Identity Key is a RSA key pair used for signing TPM structures. The AIK itself cannot sign externally (to the TPM 1.2 and below) created structures, so the AIK is used to sign a signing or general purpose key to make it a certified key to be used as a device or user identity RSA private/public key pair for general purpose authentication. The AIK key and certificate MUST be created and enrolled as specified within [21]. The AIK is required to sign some information about the authentication key. The result of that certification process is reflected in the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure.

3.1.1.2.3 *Device Identity Authentication Key Creation – Trust chain via standard certificate fields*

A Device Identity Authentication Key MUST be created as follows:

Use the AIK (section 3.1.1.2.2) with the TPM_CertifyKey function to certify a non-migratory signing or general purpose key to be used for the DevID.

The resulting non-migratory authentication key is (statistically) unique to a single TPM and cannot be migrated or exported from the TPM Version 1.2 and below.

Figure 4 shows an overview of the certificate signing hierarchy for this DevID certificate.

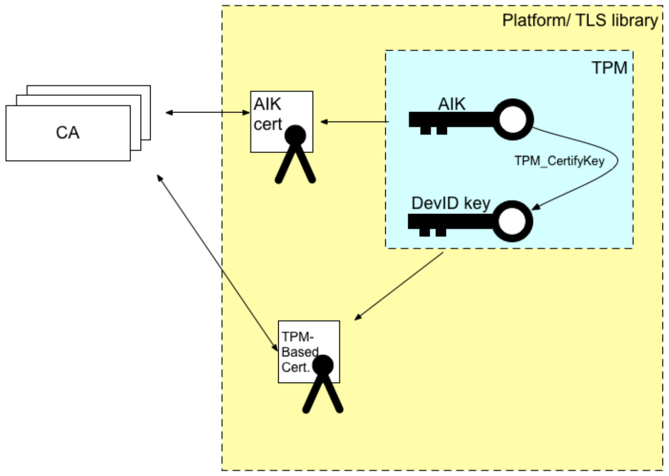


Figure 4 Certificate Chain via Certificate Fields

The next paragraphs describe the detailed steps need to create the DevID certificate.

The authentication key MUST have the following TPM_KEY_USAGE values which are a subset of the TPM_KEY_USAGE Table in Section 5.8 of the TPM 1.2 specification [48]. The key usage value determines the encryption and/or signature schemes to be used with the key.

Property	**Comment**
TPM_Key_Signing	This value is required because it supports all authentication protocols that require a signing operation. For example, Diffie Hellman based cipher suites are used (in contrast to RSA key exchange). If unsure or if RSA key exchange algorithms have to be used, use a Legacy Key instead.
TPM_Key_Legacy	Can perform signing and binding operations and may be used for both.
Non-migratory	The key must be non-migratory.

For the cases where encryption is used for authentication (as it is the case for RSA key exchange cipher suites), the key MUST be a TPM_KEY_LEGACY key.

A key usage value TPM_Key_Migration indicates that the key is migratable when set. Therefore, the TPM_Key_Migration value MUST NOT be set so that the DevID key remains non-migratable.

The TPM 1.2 (Main) Specification, Section 5.9 also contains an authorization attribute that MUST be set for the IDevID key as shown:

TPM_AUTH_DATA_USAGE – Set to TPM_AUTH_ALWAYS.

TPM_AUTH_DATA_USAGE	TPM_AUTH_ALWAYS
TPM_SIG_SCHEME	TPM_SS_RSASSAPKCS1v15_SHA1
Key size	At least 2048 bits

3.1.1.2.4 TPM Certified Keys for Alternative Trust Chains (AIK as Root of Trust Chain)

This section will describe how a TPM-based DevID that is not signed by a CA, is created and used within TLS. This section describes an alternative trust chain that can be created via the SKAE extension. The resulting TPM-based DevID will be referred to as a “TPM certified” key and certificate.

This section applies to the TLS use case mentioned in section 2.5.3.

The alternative trust chain does only confirm that a certificate is bound to a certain TPM. It does not allow to know which TPM and which platform just by checking the certificate. However the alternative trust chain ensures that this platform can be recognized again and again as the same platform.

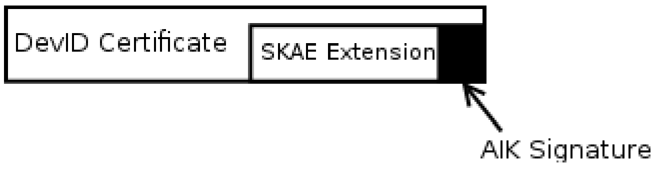
A “TPM certified” key means that the TPM creates a special structure called TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 that is signed by the TPM’s AIK. The resulting self-signed DevID includes the SKAE extension in order to carry that AIK signed TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure with it. So, the content of the SKAE extension is what binds this DevID to a TPM. It provides the proof of TPM residence. Furthermore in order to allow a verifying party to verify the AIK signature, the AIK certificate chain will be also transmitted using the TLS supplemental data handshake message.

The DevID certificate used in this case MUST contain an SKAE extension.

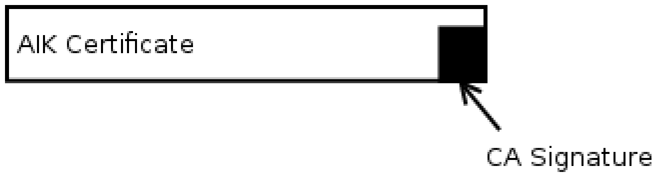
In this alternative trust chain, the AIK has been signed by a CA as usual and it has been used to verify the DevID by signing the signature over the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structures in the SKAE extension. However, the verifying party cannot validate a CA signature over the resulting DevID since it has been self-signed. Instead, the verifying party MUST validate the TPM_CERTIFY_INFO or

TPM_CERTIFY_INFO2 structure included in the SKAE and MUST receive the AIK certificate chain with the TLS handshake protocol's supplemental data handshake message as specified in section 3.3. The idea behind this is that the AIK signature over the DevID key held in the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure is considered trustworthy and no CA is needed to sign this key.

The following figures show the DevID and the AIK certificate and their trust anchors. As explained above, the AIK certificate is signed by a CA. However, the DevID certificate is self-signed and its SKAE extension is signed by the AIK. That means the DevID certificate's trust is established via an alternative trust anchor – the SKAE extension. The DevID Certificate is an X.509 certificate that includes the SKAE extension which is signed by an AIK:



The AIK Certificate is an X.509 certificate following the TCG Standards for enrollment [21] and it is signed by a CA.



The AIK Certificate is needed to verify the SKAE signature shown above.

Now, these two pieces of information (the DevID certificate and the AIK Certificate) MUST be supplied to the verifying party so that the verifying party can determine the AIK certificate is validated as well as the TPM certified key, ending up with assurance that the key created was created in a TPM and resides in a TPM. The AIK certificate is supplied via TLS in a supplemental data message during the TLS handshake (see section 3.3 for details of the handshake and extensions definitions).

For this case, the DevID certificate does not chain with the CA at all. It is self-signed. Therefore, within the X.509 certificate the authorityKeyIdentifier field is identical with the subjectKeyIdentifier field.

The DevID authentication key is created similarly to the way it is created in section 3.1.1.2.3. It is assumed that the AIK certificate has already been issued. The first step is then to create a new authentication key *K* that can be used as a client key in authentication protocols (i.e., TLS). The key MUST be created as specified in section 3.1.1.2.3.

The key will be created using standard TPM based key creation functions such as those specified in [44]. Afterwards the new key has to be certified using the AIK (e.g. using the TPM_CertifyKey function as specified in [44]). The certification function will result either in TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure. Both are equally useful for the TLS extension as described in section 3.3.

The TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure will be used to construct a SKAE extension according to [39]. Afterwards, a standard certificate library will be used to create a CSR for key *K* including the SKAE extension. The CSR will be signed with *K* itself resulting in the TPM certified key that will be used as authentication certificate. Be aware that the TPMIdentityCredentialAccessInfo field of the SKAE extension MUST be ignored for alternative trust chains.

Figure 5 shows the procedure to create a TPM certified key and highlights the components that are involved in that process.

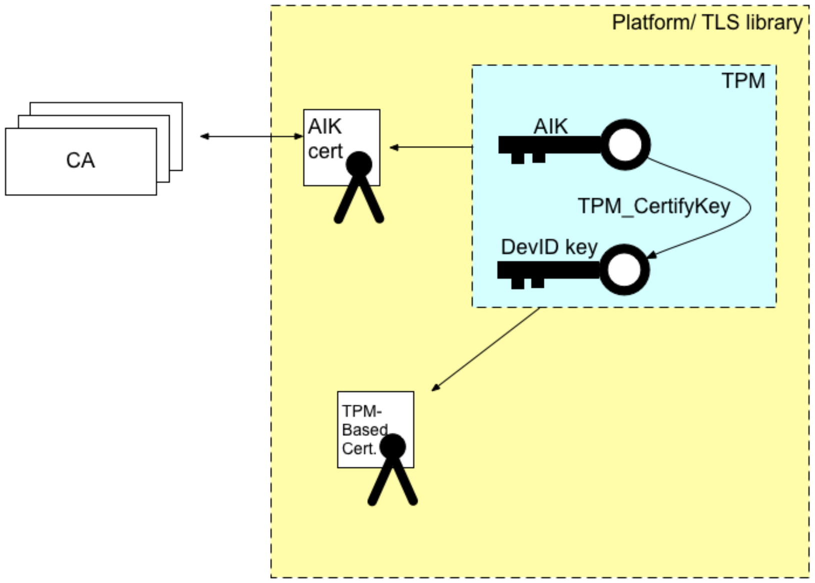


Figure 5 Creation of a TPM Certified Key

3.2 Certificate Fields in Detail

The DevID credential is a standard X.509 certificate with a profile that meets the requirements of [4] and [9]. The DevID certificate associates the distinguished name for the TPM-based identity with the authentication key used within the identified authentication protocols. This section will enumerate all of the certificate fields for the DevID credential with an emphasis on those that contain information for a TPM-based device identity DevID. In the case that the DevID certificate is created by an OEM, the subject field MUST contain a unique X.500 Distinguished Name (DN) and SHOULD include the device serial number reflected in the subject field's DN encoding as the "serialNumber" attribute (as per 802.1AR).

In this case, because the subject name field contains a value, the SubjectAltName does not need to be a critical extension but it SHALL contain a value as described below in Table 3, below which identifies the TPM as the HardwareModuleName and HwType

If a DevID is created at an Enterprise level, a subject name field SHOULD be included. As is the case when an OEM is creating the DevID certificate, the SubjectAltName SHALL contain the value as described below in Table 3, which identifies the TPM as the HardwareModuleName and HwType.

Note: If the Enterprise decides to leave the subject name field empty, the SubjectAltName extension must be critical in accordance with RFC 5280 [9].

The HardwareModuleName SHOULD be populated under SubjectAltName (OtherName within the GeneralName structure) and contain a globally unique identifier. The definition of globally unique is determined by the manufacturer or network administrator and is expected to be at least within the application domain.

3.2.1 DevID Credential Fields Summary

The DevID credential MUST contain the following subset of fields from [4]. Solutions that use the DevID credential SHOULD validate these fields as specified in [4].

Table 1 Basic Certificate Fields Utilized by TPM Based Identities

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**
Version	INTEGER	V3	
serialNumber	INTEGER	Positive Integer	Certificate serial number - MUST be a unique integer identifier for the issuing CA. Not a device serial number.
Signature	AlgorithmIdentifier Identifier	RSASSA-PKCS1-v1.5 Sha1WithRSAEncryption sha-1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }	
Issuer	Name	Name of Issuing CA	
Validity	notBefore	Date of certificate creation	In accordance with implementing organizations'

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**
	notAfter	. The GeneralizedTime value “99991231235959Z” is used for this purpose.	policies based on expected life of infrastructure certificates and risk. A DevID is expected to exist for as long as the DevID module it exists in remains operational. In order to support the standard X.509 certificate format, a certificate expiry time is specified
Subject	Name	OEMs: In compliance with 802.1AR must include Serial Number “attribute.” Enterprises: Optional	
subjectPublicKeyInfo	SubjectPublicKeyInfo	rsaEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 } Bit String (public key)	Public part of authentication key

Field Name	RFC 5280 Type	Value	TPM DevID Notes

As shown in the previous table (Table 1), the following fields SHOULD be reflected:

version

The version of the X.509 certificate. Valid DevID certificates SHALL identify themselves as version 3.

serialNumber

Certificate serial number, a positive integer of up to 20 octets. The serialNumber identifies the certificate and SHALL be created by the CA that signs the DevID certificate. The serialNumber SHALL be unique in the scope of DevID certificates signed by the CA. This will typically be different from any manufacturer serial numbers or other unique identifiers associated with the equipment in which the DevID is installed.

signature

The identity of the signature algorithm used to sign the DevID certificate.

issuer

The name of the signer of the DevID. The organization issuing the IDevID SHALL ensure that the issuer identification is unique and well-known within the field of application, for example a trademarked organization name, web address, ticker symbol, DNS domain name, etc. Because uniqueness cannot be guaranteed, the issued DevID credentials contain the authorityKeyIdentifier extension and the CA certificates in the DevID certificate chain contain the subjectKeyIdentifier extension as detailed.

validity

The time period over which the DevID issuer expects the device to be used.

notBefore

The earliest time a DevID may be used. This SHALL be the time the DevID is created.

notAfter

The latest time a DevID is expected to be used. Devices possessing a DevID are expected to operate indefinitely into the future and SHOULD use the value 99991231235959Z. Solutions verifying a DevID are expected to accept this value indefinitely. Any other value in a DevID notAfter field are expected to be treated as specified in [9].

subject

In compliance with IEEE 802.1AR, Section 7.2.8, OEM's creating DevIDs SHALL uniquely identify the device within the issuer's domain of significance. This field shall contain a unique X.500 Distinguished Name (DN). The subject field's DN encoding should include the "serialNumber" attribute with the device's unique serial number.

For Enterprise creation, the subject field is optional.

subjectPublicKeyInfo

Describes the DevID public key algorithm identifier and key value.

Table 2 Certificate Extensions Used for CA Signed Device Identity Certificates

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**
authorityKeyIdentifier	KeyIdentifier	A unique value that MUST match the subjectKeyIdentifier of the issuer's credential IF an ISSUER (CA) is Utilized. Note that in case of an alternative trust chain as described in section 3.1.1.2.4, this field is identical with the subjectKeyIdentifier field	Matching is important to support certificate chaining for DevID
subjectAltName	GeneralName otherName	type-id OBJECT IDENTIFIER value per RFC 4108: id-on-hardwareModuleName OBJECT IDENTIFIER ::= {iso(1) identified-organization (3) dod (6) internet (1) security (5) mechanisms (5) pkix (7) on (8) 4 }	HardwareModuleName ::= SEQUENCE { hwType OBJECT IDENTIFIER hwSerialNum OCTET STRING} The hwType Object Identifier is: 2.23.133.1.0 Where hwType OID = represents TPM Version 1.2 and hwSerialNum is the TPM Serial Number.
Key Usage	KeyUsage	digitalSignature and	Support for older authentication

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**
		dataEncipherment	protocols that use encryption means that the DevID keys MUST assert these uses.
Certificate Policy	Version 3 Extension	CPS Identity	The Certificate Policy extension MUST be present.
Subject Key Attestation Evidence (SKAE)	Private Extension	TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 Structure	This structure contains information describing the DevID signed by a TPM Identity Key. The SKAE extension MAY be present

AuthorityKeyIdentifier

As shown in the table above, DevID certificates issued MUST contain the non-critical authorityKeyIdentifier extension. This extension together with the "subjectKeyIdentifier" facilitates certificate path building which is necessary to validate DevID credentials. The DevID certificate MUST contain an authorityKeyIdentifier that matches the subjectKeyIdentifier of the CA certificate.

The subjectKeyIdentifier extension SHOULD NOT be included in DevID certificates.

SubjectAltName

The subjectAltName extension is a standard X.509v3 extension. The X.509 certificate profile presented in [9] specifies the subjectAltName extension for allowing the binding of additional identities to the subject of the certificate.

This non-critical extension SHALL contain a hardwareModuleName as specified in RFC 4108 [50] that describes the TPM.

The object identifier for the `subjectAltName` extension is defined as:

```
id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
```

The subjectAltName field SHOULD reflect that this identity is a device identity as described above.

```
subjectAltName = GeneralNames
otherName [0] OtherName
OtherName ::=Sequence {
    Id-on-hardwareModuleName OBJECT IDENTIFIER ::=
   iso(1)   identified-organization(3)   dod(6)   internet(1)   security(5)
mechanisms(5) pkix(7) on(8) 4
    HardwareModuleName ::= SEQUENCE {
    hwType OBJECT IDENTIFIER
    hwSerialNum OCTET STRING }
```

Note: The TCG registered OID (2.23.133.1.0) represents the hwType and TPM 1.2. The hwSerialNum is the TPM serial number. The combination of the hwType and hwSerialNum uniquely identifies the hardware module.

Key Usage

This extension MUST be included. It MUST assert that DevID keys can be used for both signing and encryption as some older authentication protocols require. This extension defines the purpose of the key contained in the certificate.

CertificatePolicy

This extension MUST be populated.

Table 4 Certificate Extensions Used for Alternative Trust Chains (see section 3.1.1.2.4)

All certificate fields should be populated as stated in Table 3 above, with the following changes:

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**
Subject Key Attestation Evidence (SKAE)	Private Extension	TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure	This private extension MUST be used if the certificate is not CA signed.
Certificate Policy	Version 3 Extension	CPS Identity	The Certificate Policy extension MAY be present but is not required for non-CA signed Device Identity Certificates.

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**

In addition:

- The content of the DevID's authorityKeyIdentifier field MUST be the same as the subjectKeyIdentifier field, since this certificate is self-signed.
- The BasicConstraint field "CA" MUST be set to false.

KeyUsage

A DevID is meant to provide a long-lived credential for identifying the device in future uses of authentication protocols. If a critical keyUsage extension is included in the DevID, it SHALL include digitalSignature as defined in [9] in support of those authentication protocols that may need to use a public key to wrap a private symmetric key. The keyUsage extension MAY include keyEncipherment

3.2.1.1 Version 3 Certificate Extensions Required for PKI trust anchors

As stated in Section 7.1 of the 802.1AR specification, to support TPM-based device identities, the credentials for PKI infrastructure components MUST conform to the table 3, above and MUST also contain the following:

Field Name	**RFC 5280 Type**	**Value**	**TPM DevID Notes**
subjectKeyIdentifier	Unique identifier associated with the public key in the certificate. Mandated by RFC 2459 for CAs		Facilitates certificate path building necessary to validate DevID credentials
BasicConstraints		CA=true	
Policy Mappings	one or more OID equivalences between CA domains		

SubjectKeyIdentifier

The non-critical subjectKeyIdentifier extension of the CA certificates MUST be populated with a unique value [9]. This value MUST contain the value placed in the key identifier field of the authorityKeyIdentifier section of DevID certificates issued by the CA. This facilitates certificate path building, which is necessary to validate DevID credentials. The DevID certificate itself does not contain the subjectKeyIdentifier extension both to conserve space and because it is not used when building certificate paths.

BasicConstraints

The Attestation CA credential SHALL include the "basic constraints" extension with CA=TRUE since the certificate is for a CA.

Policy mappings

This extension is used in CA certificates. It lists one or more pairs of OIDs; each pair includes an issuerDomainPolicy and a subjectDomainPolicy. The pairing indicates the issuing CA considers its issuerDomainPolicy equivalent to the subject CA's subjectDomainPolicy.

Note: This specification does not require the X.509 SubjectAltName extension to be critical (as stated in RFC 4108) due to interoperability issues caused when a critical extension is processed. RFC 4108 also states that the SubjectName field must be empty. These requirements are two restrictive for a DevID certificate where an end entity application must have flexibility about processing these certificates for a device identity.

3.2.1.2 SKAE Role in Certificate Request and End-Entity Verification

This section describes how the Subject Key Attestation Evidence (SKAE) and certificate policy (CP) extensions are used by the CA during the certificate request and the options for verifying party usage of this extension. This chart does not apply to the DevIDs with alternative trust chains.

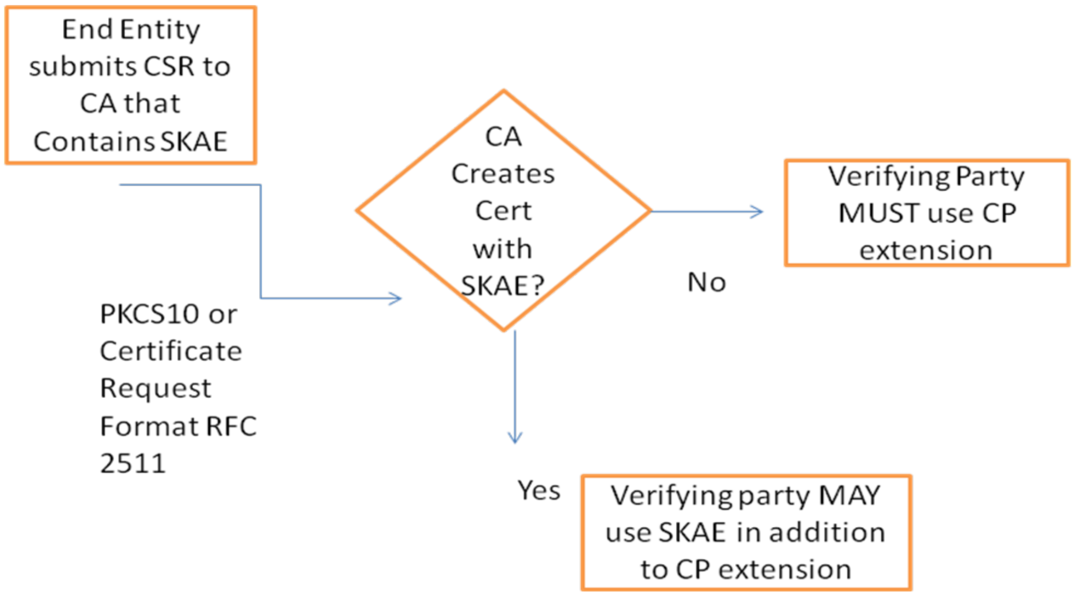

End Entity submits CSR to CA that Contains SKAE
PKCS10 or Certificate Request Format RFC 2511
CA Creates Cert with SKAE?
No
Verifying Party MUST use CP extension
Yes
Verifying party MAY use SKAE in addition to CP extension

Figure 6 Flow Chart of SKAE Usage Options

In all cases, the CSR MUST contain an SKAE. In all but the “alternative trust chain” use case (2.5.3), the CSR will be examined by the CA in order to prove TPM residence of the DevID private key. This CSR is sent to the CA via normal PKCS#10 or Certificate Request Message format [22]. The CA can then issue a DevID X.509 v.3 certificate with the optional SKAE extension and the mandatory policy extension

The SKAE extension MAY be added by the CA to DevID certificates it creates to allow verifying parties to validate the TPM origination of the DevID keys

In all cases except the “alternative trust chain” use case, the verifying party MUST determine TPM origination via a certificate policy extension which is described above and is the mandatory method of TPM residency validation.

Note: A certified key certificate signing request could be combined with a CMC-based AIK enrollment request, thus obtaining both an AIK certificate and a device identity certificate at the same time.

Based on the 802.1AR standard, DevID secrets SHALL be stored confidentially and not available outside the module. Encryption techniques MAY allow storage space that is external to the module to be used. The DevID module service interfaces MUST be used to access credentials and secrets so stored.

3.3 TLS Extensions required to use multiple certificates within one TLS handshake

TLS comes with the possibility to extend the handshake in a backwards compatible way as stated in [5]. This is done by extending the TLS hello messages. However, TLS extensions are not meant to allow for new cipher suites etc. One common example for a TLS extension is the server name extension that allows the client to tell the server the domain name it used to contact the server. The general format of the TLS hello extension as specified in [5] is the following:

```
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
}
```

The extension can be used to either transmit a small amount of data (like the server's domain name) or to signal the transmission of data within the TLS supplemental data handshake message, which will be explained below. If the client makes use of an extension, the server has to acknowledge that extension if it supports it. If not, this handshake will stop right after the hello messages. It is up to the communication partners (and the application) whether they start a new handshake without those extensions or not.

The TLS supplemental data handshake message is defined in [7]. It allows sending arbitrary data during the TLS handshake. Data that will be sent using the supplemental data handshake message has to be announced using a TLS extension.

3.3.1 TPM certified keys

There are cases that want to make use of the AIK certificate for authentication purposes (e.g. see use case 2.5.3). However the AIK itself cannot be used for authentication within TLS since it signs TPM originated data only. There have been several ideas to circumvent that problem, each with their disadvantages:

1. Use the AIK to sign a general purpose key that can be used within TLS: This is not possible due to the "CA:false" constraint as specified in [43].
2. X.509 proxy certificates as specified in [9]: One could use the AIK certificate as proxy issuer certificate, however, according to [9] it is not allowed to have an empty subject field within the proxy issuer certificate. [43] Specifies the AIK certificate's subject field as empty.
3. Let the TPM create a new certified authentication key. Furthermore, create a CSR for that key including the SKAE extension as specified in [39]. This CSR needs to be signed by a CA and the resulting certificate will then be used as TLS client certificate. Although this scenario will work smoothly with a lot of implementations, it increases the costs at the identity provider's side due to the CA request required for the certified key.

In order to overcome the issues described above, this section assumes that a platform has a TPM certified key certificate as described in section 3.1.1.2.4

This TPM certified key will now be used as TLS identity certificate. However the AIK certificate is needed as well in order to be able to verify that the key K is a TPM protected key. The TPM certified key can be transmitted using the standard TLS handshake messages. The AIK certificate needs to be transmitted using TLS extensions as defined in [5] combined with the TLS supplemental data handshake message as defined in [7]. The basic idea is to use a TLS extension to signal the use of a TPM certified key and send the AIK certificate within the supplemental data handshake message.

The new extension types for a TPM enabled TLS client or server are called client_devid and server_devid:

```
enum {
   client_devid(TBD), server_devid(TBD), (65535)
} ExtensionType
```

This extensions MAY be used in full handshakes as well as in session resumption handshakes. Although the latter does not require a certificate exchange it might happen that the server refuses to accept a resumed session and runs a full handshake instead. In order to be able to do that without interruption, the extensions SHOULD be included also in the session resumption handshake.

The client sets client_devid in case it plans to use a TPM certified key. If the client wants the server to use a TPM certified key, server_devid has to be set.

The following combinations of client_devid and server_devid are possible:

1. Client_devid set, server_devid not set
2. Client_devid not set, server_devid set
3. Client_devid set, server_devid set

After having announced that one or both sides will make use of TPM certified keys, the client or server that actually uses such a certificate has to supply the AIK certificate within the supplemental data handshake message.

The supplemental data handshake message to be used to send the AIK certificate is defined as follows:

```
enum {
   tpm_certificate(TBD), (65535)
}
```

with

```
struct {
    SupplementalDataType supplemental_data_type;
    select(SupplementalDataType) {
       case tpm_certificate: TPMCertData;
    }
} SupplementalData
```

and

```
opaque ASN.1Cert<2^24-1>;

struct {
      ASN.1Cert certificate_list<0..2^24-1>;
} TPMCertData;
```

In order to prove that the entity does not only possess the AIK certificate, but also the AIK private key, a proof-of-possession (POP) is needed. This POP is given with the SKAE extension of the TPM certified key certificate that is used as TLS client certificate. Summarizing, the SKAE serves two purposes here:

1. Provide a POP for the AIK. Since an AIK does only sign TPM originated data, a POP has to be provided in an indirect way: The SKAE extension has been signed using the AIK. If the verifying party is able to verify the SKAE extension of the TPM certified key certificate, it has proven that the client possesses the private part of the AIK as well. Furthermore as specified in section 3.4, the verifying party SHOULD make sure that the AIK certificate has not been revoked.
2. Provide a TPM residence proof for the TPM certified key. According to [44], the AIK certifies non-migratory keys only. Since the SKAE extension carries the TCPA_CERTIFY_INFO or TCPA_CERTIFY_INFO2 structure that has been signed by the AIK, it is proof, that the key used as TLS client key is a TPM based non-migratory key.

TPMCertData carries the entity's AIK certificate. This SHOULD include the complete chain.

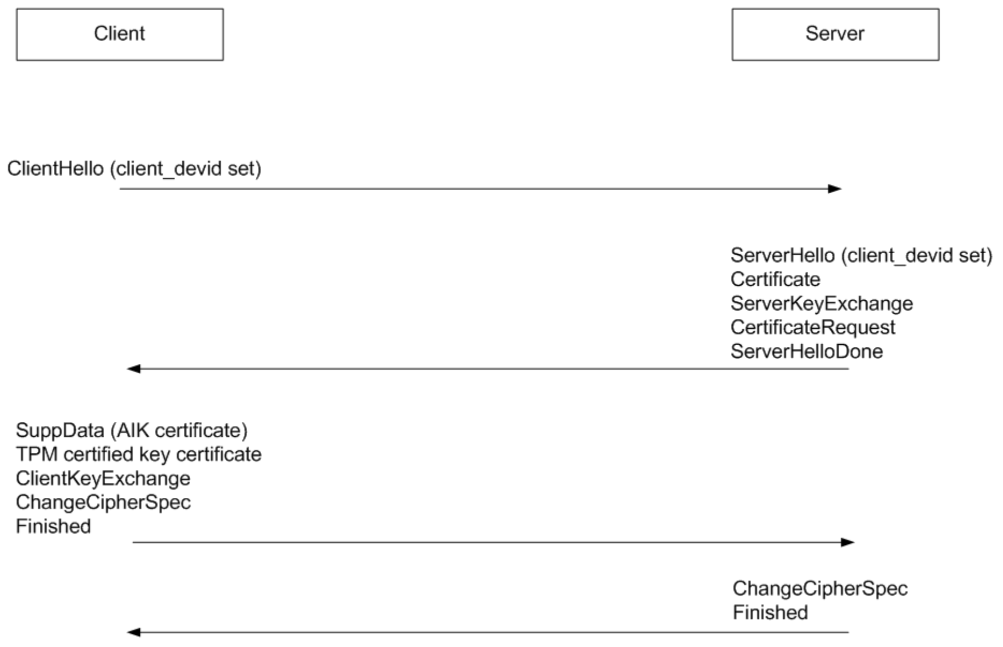


Figure 7 Full TLS Handshake with Client Using a TPM Certified Key

Figure 7 shows a full TLS handshake where the client uses a TPM certified key. The handshake starts with the ClientHello message including the client_devid extension in order to prepare the server for those special certificates. Since the server supports TPM certified key certificates, it confirms it by including the client_devid extension in the ServerHello message. The server also includes its own standard X.509 certificate within the Certificate message and sends the standard TLS ServerKeyExchange message. Since the client needs to be authenticated, the server sends the CertificateRequest message (knowing already that this will result in a TPM certified key certificate) and concludes with the ServerHelloDone. Afterwards it is up to the client to send its AIK certificate within the SuppData message as well as the TPM certified key certificate as TLS client certificate. The client will conclude sending the ClientKeyExchange, ChangeCipherSpec, and Finished messages. The server will then conclude the handshake with ChangeCipherSpec and Finished.

As shown in Figure 7, the TLS hello extensions and the supplemental data handshake messages are also protected by the Finished message and are therefore integrity protected.

3.4 Verifying Party Requirements

The applications will verify the digital signature and parse the device identity certificate in the following manner:

When a verifying party receives a DevID certificate, processing is as follows:

1) If the DevID is signed by a CA, the verifying party MUST verify the CA signature and

MUST parse the certificate policy extension field to verify that field contains the TCG policy OID as evidence of binding between the DevID and the TPM.

a) The verifying party MAY also parse the SKAE X.509 extension.

2) If the DevID TPMCertData extension is used (that means that the alternative trust chain is used), the verifying party MUST verify the AIK certificate chain and the TPM certified key certificate including the SKAE extension as for instance done in use case 2.5.3.

Figure 8 presents the requirements for the verifying party as decision tree.

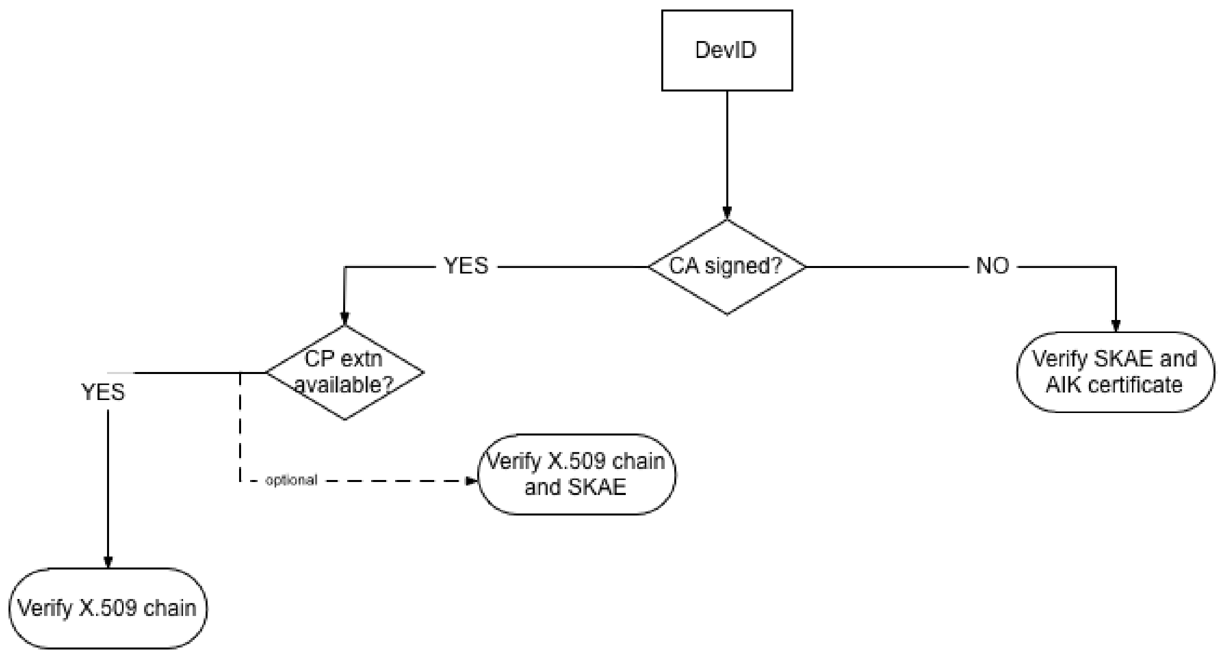


Figure 8 Decision Tree for the Verifying Party

4 TPM-based Device Identity Lifecycle Requirements

4.1 Provisioning

TPM based DevIDs are expected to have a long-life. If Integrity of the DevID is to remain from factory to operational use, the factory-installed DevID MUST NOT be removed. The initial provisioning and registration processes are important to the integrity and security of the device identity. For this reason, the DevID SHOULD be created in conjunction with the Endorsement Key Certificate and the AIK Certificate, both of which SHOULD be enrolled in accordance with applicable (EK or AIK) TCG enrollment specification [21](REF)..

At time of provisioning, an organization SHOULD determine if there is a need to create back-up copies of the DevIDs in order to recreate the these keys in the event of TPM or motherboard failure. This will preserve DevID functionality in the event of hardware failure.

4.2 Maintenance

Standard X.509 Public Key Infrastructure standards apply to maintaining the DevID certificate and keys.

5 Security Considerations

5.1 Threat Model

T1. An adversary will attempt to access the DevID private key in order to spoof the identity of the device.

T2. An adversary will attempt to attack the private key or interfere with the creation of the DevID during the time of its creation or by inserting false information into the provisioning infrastructure resulting in weak or unprotected EK, AIK or DevID keys and certificates.

T3. OEM manufactures that do not provide unified trust certificates provide no assurance that a TPM is initially correctly attached to the equipment motherboard or evidence of a certified trusted building block (TBB). TPMs could be physically attacked or substituted with a bad or degraded TPM in this case.

T4. An adversary may attack vulnerabilities or incorrect implementations of well-known authentication protocols in order to masquerade as a good device. Alternately, an adversary may use protocol vulnerabilities or implementation weaknesses to launch a man-in-the middle (MITM) attack, replaying the authentication protocol containing the DevID.

T5. Commands to and APIs used in conjunction with the TPM could be modified or broken to result in the incorrect operation of the TPM.

T6. A denial of service (DOS) attack could be launched and physical access to the TPM was achieved. This DOS attack could be accomplished by clearing the TPM ownership which causes the loss of existing keys protected by the TPM.

T7. In the case where the same secret auth_data is used for several DevID keys. An attacker could gain this knowledge and trick the TPM to load the wrong key, authenticating it versus the intended DevID key [49].

T8. If two separate network protocols are used with one containing the user certificate and a second protocol containing the device certificate, insecurities can appear if the two certificates are not bound together or associated in a secure manner. Some databases do IP address indexing which is weak.

5.2 Countermeasures

C1. Countermeasure for Access to Private Key:

The private key is to be stored securely which is accomplished by the TPM. In order to prevent an adversary from accessing the private key and spoofing the device identity, the key is cryptographically protected and bound to the TPM and can only be decrypted within the protected execution environment of the TPM.

C2. Countermeasure for Attack to Keys and Certificates During Provisioning

Use the referenced TPM enrollment protocols within this specification. Generally, the less distributed the enrollment process is, the more secure it is. In other words, if the EK, AIK and DevID are all created at one time by the trusted infrastructure at a manufacturing or OEM plant, there is less opportunity for manipulation of the TPM keys and certificates. However, doing enrollment at such an early time is often impractical; if the key enrollment is done at a later state (e.g. in order to make sure that the TPM comes un-owned), the environment and processes for the enrollment is expected to be controlled and monitored.

C3. Countermeasure for Physical TPM Attack

The original equipment manufacturer (OEM) can create a Unified Trust Certificate which contains an assertion by the system manufacturer that the TPM is properly certified and incorporated into the platform and certification of the trusted building block (TBB). Alternately, the manufacturer, or using organization or customer can create a statistical measurement process or a conformance evaluation to determine security baseline and perform quality control measures on the physical TPM.

C4. Countermeasures for Protocol Attacks

Authentication protocols are expected to be implemented in accordance with their respective RFC and any applicable published security implementation guidance. For tunneled EAP methods, there is good MITM (Man-in-the-Middle) countermeasure guidance provided by [42]

C5. Countermeasure for TPM API Attacks:

Protect commands involving sensitive data fields (e.g. usage auth values and APIs to the TPM) by way of a trusted software interface (for example, the TSS) that provides confidentiality of the sensitive data fields and Integrity of commands made to the TPM. Elements of this countermeasure are execution environment integrity and attestation of the platform. Another integrity countermeasure is provided because authorization of commands is done via an HMAC over the command + parameters + random number + auth_data.

C6. Countermeasure for DOS Attack:

Owner_auth is set upon establishing TPM ownership. Owner_auth provides protection against a remote adversary doing a DoS attack via a software reset. There is a method to clear the TPM ownership by touching the BIOS and this can also be disabled if desired. If the TPM is physically reset, a new DevID can be issued. It is a good practice to backup any <u>user</u> keys that are considered important since they secure keys.

C7. Countermeasures for attack on auth_data:

If the same secret auth_data is used on a DevID key and a non-DevID key, this wrong key cannot be misused. However if two valid DevID certificates are loaded, one could cause the authentication with the wrong DevID. It is therefore strongly recommended to use different secret auth_data for all keys.

Furthermore it is recommended not use to the well-known SRK secret (all nulls) or if using this all null SRK secret, at least to establish an encrypted transport session with the TPM on top of the normal TPM protocols in order to avoid an adversaries ability to observe the communication used to generate the AIK or the DevID (which is a child of the SRK) and thus learning the child keys usage auth.

C8. Countermeasures for Weak Binding of User and Device Certificates

A form of channel binding could be used to link these certificates or, a strong linkage mechanism such as a URI or hash can be utilized within the database.

6 Privacy Considerations

These unique identifiers are meant to be long-term identities for end user or infrastructure devices. It is therefore important to use them only when needed. That means, make sure you really want to run a device based authentication and that the authentication server is really allowed to identify your device. Using the DevID, a special device can be uniquely identified, however this is the goal of authentication.

When network confidentiality is a concern it is proposed to run an anonymous authentication (the client authenticates the server but not vice versa) to set up a secure tunnel. Further authentication where a DevID is transmitted will then run over this secure tunnel in order to preserve confidentiality.

It might be possible to gather a lot of data about a device and/ or a user when the same identity is used for different applications. It is therefore proposed to use a DevID only for one application. When following this specification it is possible to request more than one AIK certificate to create more than one authentication key. This allows the use of pseudonyms for different application and does therefore allow to preserve the privacy of the device and user.

This specification talks only about DevIDs for reasons explained in section 2.2. If one wants an IDevID using a TPM, another privacy problem might arise. Creating a DevID requires to take ownership. In order to create an IDevID, the manufacturer needs to take ownership of the TPM. However that means that ownership cannot be re-claimed on premise. That would destroy the original IDevID. In order to keep the IDevID, the manufacturer would have to hand over the owner password in order to allow a company to own their hardware completely. However that means the IDevID is not trustworthy anymore. There is no solution to that problem on TPMs prior to and including version 1.2.

7 Examples and Use Cases

7.1 Network operator-related thoughts

In network operator scenarios (e.g. in PWLAN authentication as indicated in use case 2.5.3) it is necessary to bind customers (and their devices) to a contract in order to charge them usage/ time/ flat based. There are several ways to do that:

1. Operators usually have shops, where customers buy their hardware (even notebooks).
    1. Pre-provision those notebooks (according to one of the methods specified above), store an ID within a database and put an RFID/ barcode/ … onto the device
    2. If a customer buys a device, the RFID/ barcode / … will be scanned and the customer bound to the device.
    3. Revocation is needed in case the customer sells/ loses the device (e.g. via the operator's hotline)
2. In case the customer bought her device somewhere else, she might
    1. Go to the operator's shop and retrieve an AIK certificate via a dedicated line/ hotspot. Since it is a dedicated line/ hotspot, the operator will be able to grab the AIK and bind it to this customer (the operator might even run the CA itself)
    2. If she is already registered with the operator (e.g. has a TV, wireline contract…), she could access a web portal and retrieve and register the AIK certificate via that portal.

All the operator AIK certificate retrieval procedures will work as described in the general intro for certificate setup. It is just the user interface that will be different.

7.2 Use Cases and Requirements Recap

Use Case	**Covered in section…**
Single Authentication Certificate: Client-side Device Identity (2.5.1)	7.2.2 Using TPM Keys in IKE (within IPSec) 7.2.3 Using TPM Keys in PKINIT 7.2.6 Using TPM keys with TLS for device authentication
Single Authentication Certificate:	7.2.2 Using TPM Keys in IKE (within

Server or Infrastructure Component (2.5.2)	IPSec) 7.2.4 Using TPM Keys in EAP for device authentication 7.2.6 Using TPM keys with TLS for device authentication
Single Authentication Certificate: User/Device Authentication in Public Wireless Networks (IEEE 802.1X/ EAP-TLS) Using TLS Extensions (2.5.3)	3.3.1 TPM certified keys
Device and User Identity Certificates within two Separate Network Security Protocols (2.5.4)	7.2.8 Using TPM-based Identities in Multiple Network Protocols
Device and User Identity Certificates within Tunneled EAP (2.5.5)	7.2.5 Using TPM Keys in Tunneled EAP for user and device authentication 7.2.7 Using TPM Keys with IF-T
Device and User Identity Certificates within TLS using (2.5.6)	7.2.7 Using TPM Keys with IF-T
Device and User Identity Certificates within one authentication protocol (2.5.7)	7.2.2 Using TPM Keys in IKE (within IPSec) 7.2.1 Using TPM Keys in SASL EXTERNAL-TLS

7.2.1 Using TPM Keys in SASL EXTERNAL-TLS

The SASL Mechanism Family for External Authentication EXTERNAL-* as defined in [19] allows two authentication partners to define EXTERNAL-TLS explicitly as authentication mechanism to be used. Furthermore, EXTERNAL-TLS as defined in [19] allows to run several chained TLS handshakes. The first TLS handshake is a standard handshake. Successive handshakes will actually be renegotiated handshakes. All of the chained TLS handshakes are cryptographically bound together using the safe renegotiation extension of TLS [20].

If only one authentication certificate is used (be it a TPM based key or not), one can just use SASL EXTERNAL-TLS as is.

When using TPM based device and user certificate with the SASL EXTERNAL-TLS, it is proposed to use the DevID in the first handshake and the user certificate in the second, renegotiated handshake. The verification of the certificate MUST be done as specified in section 3.4. If one of the two verifications fails it is up to the application if access will be granted anyways or not. However it is recommended that only limited access will be granted. If both verifications fail it is recommended that access will be denied.

7.2.2 Using TPM Keys in IKE (within IPSec)

This section describes how the TPM-based DevID certificate can be used within the Internet Key Exchange (IKE) protocol that operates within IP Security (IPsec) to provide confidentiality, data integrity, access control and data source authentication to IP datagrams. TPM-based DevID certificates can be used within both IKE Version 1 and IKE Version 2 to support authentication of an endpoint. Implementations of each are expected to meet the appropriate RFC. [16][17]

IKE performs mutual authentication and establishes an IKE Security Association (IKE SA) which in turn installs one or several IPsec or Child SAs using either the Encapsulating Security Payload (ESP) or Authentication Header (AH) IPsec protocols. This specification will address how to use TPM-based certificates within IKE authentication.

Using IKE with the TPM-based DevIDs meets the following use cases:

- Single Authentication Certificate: Client-side Device Identity
- Single Authentication Certificate: Server or Infrastructure Component
- And it can be used together with another security protocol to meet “Device and User Identity Certificates within two Separate Security protocols”.

7.2.2.1 IKEv2

The two components defined by the IKEv2 RFC that are relevant to this specification are the Identification Payloads (IDi and IDr) and the Authentication Payload (AUTH).

As shown below, the IDi and IDr Identification payloads allow for identity assertions and for the purpose of this specification, for the inclusion of the DevID in the IDi and IDr. The identification payload consists of the IKE generic payload header followed by identification fields. The ID Type (1 Octet) according to the RFC will be ID_DER_ASN1_GN which is the binary DER encoding of the ASN.1 X.509 subjectAltName [9]. The subjectAltName is constructed as stated in section 3.2.1.

The Authentication payload for the purposes of this specification is a RSA Digital Signature (using SHA-1 as the hash function for the authentication payload signature.)

Pursuant to the IKEv2 RFC, to use IKEv2, the following two major steps take place:

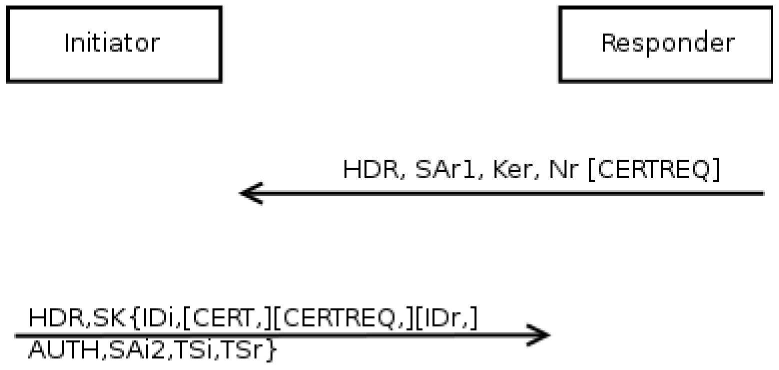
1. IKE_SA_INIT (to negotiate algorithms and establish keys for the IKE Security Association by exchanging nonces and DH values)

2. IKE_AUTH – authenticates the previous messages, exchanges identities and certificates, and establishes IKE_SA and the first Child Security Association (CHILD_SA).

The first request/response message pair performs an ephemeral Diffie Hellman key exchange and the negotiation of cipher suite parameters. In the second message exchange the initiator proves its identity by sending a digital signature in an AUTH payload. This is where the DevID private signing key is used. The initiator then provides the DevID certificate as shown below in the CERT field as reflected in [17]. It is recommended that the method used to access the DevID credentials be a standard and secure method such as PKCS#11. It is recommended that the credentials be stored securely within the TPM or can be stored securely (encrypted) off the TPM. (See section 5 for additional information).

IKE_SA_INIT

The SA_INIT response is shown below as this is the first place that a certificate can be requested for use to authenticate a device:

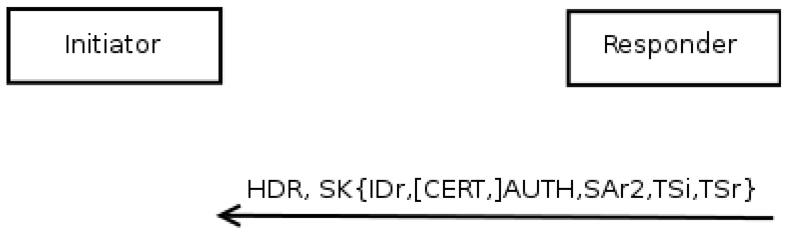

Initiator
Responder
HDR, SAr1, Ker, Nr [CERTREQ]
HDR,SK{IDi,[CERT,][CERTREQ,][IDr,]
AUTH,SAi2,TSi,TSr}

The notation SK { } indicates that these payloads are encrypted and integrity protected.

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload.

The initiator also sends a list of its trust anchors in CERTREQ payloads(s). The CERTREQ payload is optional and dependent on which party needs to request a certificate first. The first certificate provided will contain the public key used to verify the AUTH field, per [17]. The optional payload IDr (as shown) enables the initiator to specify which of the responder's identities it wants to talk to. This is useful when the machine on which the responder is running is hosting multiple identities at the same IP address.

The initiator begins negotiation of a CHILD_SA using the SAi2 payload. The final fields (starting with SAi2) are described in the description of the CREATE_CHILD_SA exchange within [17].



To use a DevID certificate within IKEv2, the digital signature based authentication method is used as described in [17]. The algorithm used is based on RSA X.509 certificates containing RSA public keys as stated above in section 3.1.1.2.3.

The verifying party then performs validation of the DevID certificate as stated in section 3.4 above.

7.2.2.2 IKE Version 1

As referenced in RFC 2409 [18], IKEv1 differs from IKEv2 in that the authentication takes place during phase 1, as a part of the initialization not in a separate IKE_SA_INIT/IKE_AUTH exchange as described above.

Using IKEv1 with TPM-based DevID certificates satisfies the use cases for a single authentication certificate, client side device identity (section 2.5.1) and a single authentication certificate for a server or infrastructure device (section 2.5.2).

IKEv1 is predominately implemented using digital signature authentication; the RFC also allows for two forms of authentication with public key encryption or a pre-shared key authentication option. This specification supports the usage of TPM keys and certificates as the DevID and uses digital signature encryption. Authentication with public key encryption is out of scope for this specification because of its limited use. Therefore, to use DevID certificates within IKEv1, use the main mode and perform authentication using digital signature.

In IKEv1, the authentication payload is designated as a SIG_ or SIG_R. This payload is the result of the negotiated digital signature algorithm applied to HASH_I or HASH_R, respectively. The certificate payload remains designated as CERT.

IKEv1 in Main Mode consists of 3 pairs of messages. The first is where IKEv1 security policies are configured, the second is the Diffie Hellman Key exchange, and the third is the authentication using digital certificates.

Following the RFC’s description of Main Mode, after the Header, version numbers and cryptographic algorithms are negotiated and nonces are exchanged, the following takes place for the digital signature authentication:

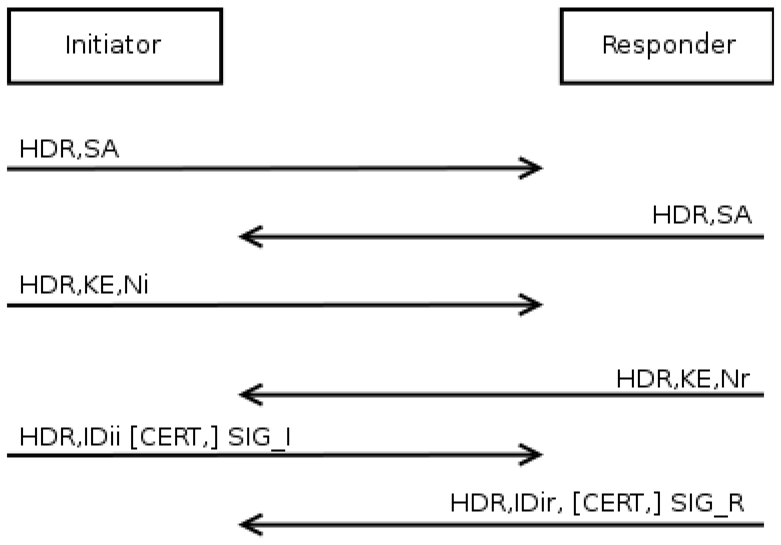


Figure 9 IKEv1 Authentication with Digital Signatures

HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. IDii and IDir are the Initiator and Responder identities, the SIG_I or SIG_R is the result of the negotiated digital signature algorithm applied to the hash payload. The DevID certificate is placed in the CERT field. The verifying party then performs validation of the DevID certificate as stated in section 3.4 above.

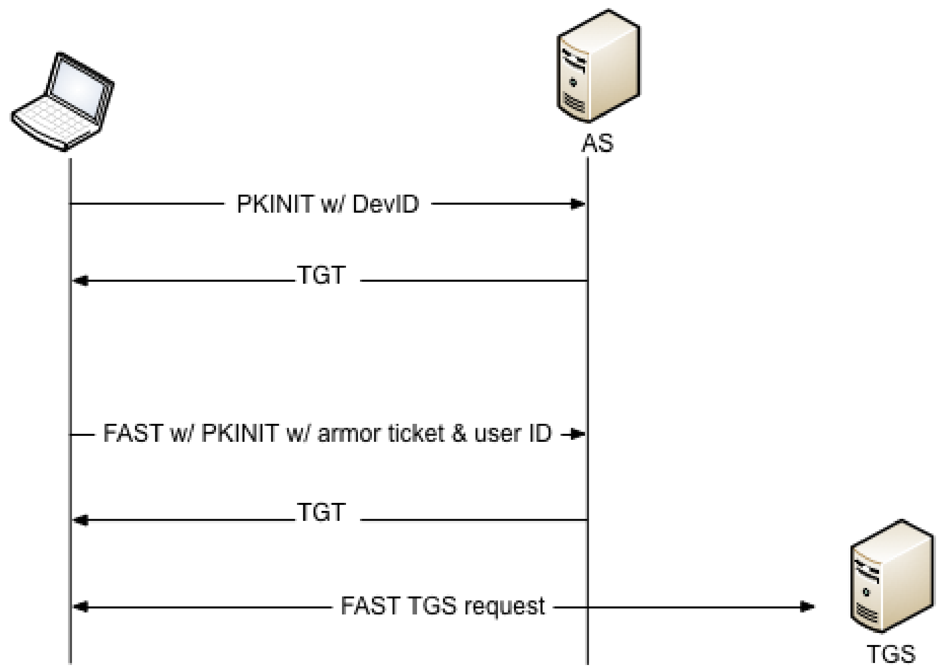
7.2.3 Using TPM Keys in PKINIT

Kerberos is a protocol to verify the identity of so called principals. A principal in Kerberos terms is "a uniquely named client or server instance that participates in a network communication" [33].

The original Kerberos services as defined in [33] did not include any pre-authentication mechanisms. They have been in added later and one of those added pre-authentication mechanisms has been defined in [34]. Pre-authentication is needed "to prove the identity of a principal" [34] that is used to authenticate before it sends the first ticket request to the Authentication Server (AS). Furthermore, some pre-authentication mechanisms allow encrypting tickets with a key different from the long term key.

One way to do pre-authentication is based on certificates and called PKINIT. The "Generalized Framework for Kerberos Pre-Authentication" [34] defines a general framework on how to use those pre-authentication mechanisms in a secure way. Amongst others it defines FAST, which is short for "flexible authentication secure tunneling". Compared to PKINIT, FAST allows securing much more elements of the Kerberos protocol messages. In general, FAST specifies what to do with padata (short for "pre-authentication data"), which is a typed hole in the original Kerberos protocol [33]. FAST also defines so called FAST armors which are keys to be used to encrypt the padata.

The next paragraph discusses the details of applying this spec to PKINIT and FAST based on the following figure:



As mentioned, the client requests a TGT at the Authentication Server (AS). This is done using a PKINIT request with the DevID certificate. If the client would send the PKINIT request without the device certificate, the AS would reply with KDC_ERR_PREAUTH_FAILED. Furthermore, the AS could indicate the pre-authentication mechanism it accepts so that the client can resend the TGT request. After having received the TGT request, the AS has to verify the client's device certificate. If this verification is successful, the AS will reply and send back a TGT as specified [35].

Since the KDC requires the authentication via user and device certificate and since the client did only provide the device certificate so far, another TGT is requested with the help of the user certificate. This is done using a FAST request. Furthermore, that FAST request carries PKINIT with the armor ticket derived out of the first TGT request plus the user certificate. The whole request will be encrypted under the so called FAST armor. This FAST armor is basically just a key created out of the first TGT, the session key between KDC and client, and a subkey. The subkey has been defined in [33] as optional random string chosen by the client. It is mandatory with FAST. This FAST armor realizes the cryptographic binding between the first request and the current request. Therefore it does also provide the cryptographic binding between the device and the user certificate. If the AS is able to verify the request, it will release the next TGT which can then be used to request a service ticket at the TGS.

In order for a kerberized service to be able to know which DevID was used to authenticate the user, special authorization data is used. The authorization data carries the principal name of the DevID.

In general, the Kerberos principals are generated out of the SubjectAltName of the DevID and the Subject or SubjectAltName of the user certificates. The SubjectAltName of the DevID carries the HardwareModuleName and the hwType OID, 2.23.133.1.0 together forming the principal of the DevID. The Subject of SubjectAltName field of the user certificate will most like have a CN (Common Name in X.509) set. Therefore, the authorization string will have the following form:

“devid”+”.”+OID+”.”+hash+”.”+”userid”+”.”+CN@REALM

It is up to the application to define the realm.

7.2.4 Using TPM Keys in EAP for device authentication

This section is about how to use device authentication based on TPM keys within EAP. In order to use user <u>and</u> device authentication at the same time within (tunneled) EAP, refer to section 7.2.5.

The DevID as defined in this specification is a standard X.509 certificate with certain specialties regarding the verification. Therefore, in order to use the DevID in EAP, a certificate based EAP method has to be used. Examples are EAP-TLS [25], EAP-FAST [26], or PEAP [27]. The DevID will be used as client certificate for those EAP methods. When verifying the certificate, the EAP authentication server has to follow section 3.4.

7.2.5 Using TPM Keys in Tunneled EAP for user and device authentication

Use case 2.5.5 describes a scenario where a TPM based user and a device certificate are used within tunneled EAP. There are several options to implement this use case:

- Use a certificate as specified in section 3.2 combined with chained EAP methods.
- Use a certificate as specified in section 3.2, where one of the two certificates is used as client certificate for the tunnel method and the second as client certificate for the inner method.
- Use chained TLS handshakes cryptographically bound to each other using the safe-renegotiation extension of TLS.

7.2.5.1 Chained EAP methods without TLS handshake chaining

In order to use a TPM based user and device certificate with two chained EAP methods, TEAP [30] is used as outer authentication method. If the tunnel has been set up successfully the first inner method is going to start. This has to be a certificate based authentication methods such as EAP-TLS [25]. It is up to the application to decide if the device or the user certificate will be used first. However it is common to use the device certificate first.

The following table shows the different options based on the result of that first inner authentication.

Result of first inner authentication	**Follow Up**	**Result of second inner authentication**	**Follow Up**
SUCCESS	Go on with 2nd method	SUCCESS	Client is authenticated
FAILURE	Go on with 2nd method	SUCCESS	Client gets limited access
FAILURE	Go on with 2nd method	FAILURE	Client is not authenticated
FAILURE	Stop authentication and set client to unauthenticated		
SUCCESS	Go on with 2nd method	FAILURE	Client gets limited access

In order for the verifying party to determine whether an authentication was successful or not, it has to follow the rules in section 3.4.

7.2.5.2 Tunneled EAP without TLS handshake chaining and without inner EAP method chaining

Instead of using chained EAP methods for inner authentication, one could also use the user or device certificate as client certificate of the tunnel method, whereas the second certificate is used as client certificate for the (one) inner method. The following table shows the different options based on the results of the authentication methods.

Result of tunnel method	**Follow Up**	**Result of inner method**	**Follow Up**
SUCCESS	Go on with inner authentication	SUCCESS	Client is authenticated
SUCCESS	Go on with inner authentication	FAILURE	Client gets limited access
FAILURE	The tunnel cannot be set up and client will therefore be set to unauthenticated		

In order for the verifying party to determine whether an authentication was successful or not, it has to follow the rules in section 3.4

7.2.5.3 Tunneled EAP with EAP-TLS and TLS handshake chaining in the inner authentication method

If EAP-TLS is going to be used as inner authentication method, one could also use a TPM based user and device certification with chained TLS handshakes. The verifying party requirements for this case are described in section 3.4.

7.2.5.4 Tunneled EAP with client authentication in EAP-TTLS

Although it is usually not used, EAP-TTLS allows for certificate based client authentication. If the TLS-TPM extensions are supported, one could send a TPM based user and device certificate with chained TLS handshakes.

7.2.6 Using TPM keys with TLS for device authentication

The DevID defined in this specification is a standard X.509 certificate possibly with the SKAE extension and some special requirements for verification. Therefore when TLS is used to authenticate the device, this X.509 certificate can simply be used as TLS client certificate. In order for the server to verify the certificate, it has to follow section 3.4.

7.2.7 Using TPM Keys with IF-T

The IF-T protocol is the TNC transport protocol used to transmit IF-TNCCS messages. The following sections describe how to use TPM keys specified in this document in the IF-T bindings for tunneled EAP and TLS.

7.2.7.1 IF-T Protocol Bindings for Tunneled EAP Methods

The IF-T protocol bindings for tunneled EAP methods are specified in [14]. The specification describes how to use EAP-TNC over tunneled EAP either using EAP-TNC as single inner method or as one inner method within a sequence of inner EAP methods.

EAP-TNC itself does not provide any authentication. Therefore in order to use EAP-TNC in a tunnel method, refer to section 7.2.5 for examples.

7.2.7.2 IF-T Protocol Bindings for TLS

The IF-T protocol bindings for TLS are specified in [15]. This specification describes how to use TLS to transmit IF-TNCCS messages. The requirements specified in section 2.7 mention that TPM based keys as specified in this specification will work together with the IF-T TLS binding. This is easy given IF-T makes use of standard TLS. It is therefore possible to apply the TLS extension specified in section 3.3 directly to IF-T bindings for TLS. The requirements for the verifying party as specified in section 3.4 will be applied.

The IF-T bindings for TLS also mention the possibility to use SASL. Examples for SASL in TLS can be found in section 7.2.1.

7.2.8 Using TPM-based Identities in Multiple Network Protocols

Two network protocols can support the passing of a device certificate and a user certificate. For example, a layer 2 tunnel can be established with 802.11i using EAP-TLS (Extensible Authentication Protocol-Transport Layer Security) [25] to authenticate the device. The endpoints can be a WLAN client and an access point or a WLAN client and a switch, depending on implementation choices made. The access point requires authentication of the client to an Authentication Server usually with the access point acting as a pass-through device. EAP-TLS messages traverse via 802.1X usually over RADIUS or another mutually supported secure channel. The Client and Authentication server exchange device identity certificates (DevIDs) as a part of the normal EAP-TLS exchange.

Subsequently, an IPsec tunnel can be established at Layer 3 between the client and server. This IPsec tunnel can use IKE which will incorporate a client user certificate for the purpose of user authentication.

If desired, mutual authentication of both user and devices can be accomplished.

The binding of the TPM-based device identity and the user identity is performed within a Metadata Access Point (MAP) server per the TCG IF-MAP [41] specifications or via another secure back-end database

If desired, access decisions can be incremental. Initial decisions about what level of network access will be granted can be made based on the device identity and the subsequent user authentication can determine the correct access levels to be granted to information and data stores. Or, access decisions can be made on the presence or absence of both certificates in the secure database.

Result of tunnel setup (1st certificate) and Authentication	**Result of Tunnel setup (2nd Certificate) and Authentication**	**Follow Up**
Successful	Successful	Communication

		continues with both user and device authentication completed
Successful	Failed	Only one authentication has been successful, limited access based on whether device or user has been authentication.
Failed	Failed	The client will be rejected. Since authentication has not been achieved, neither protocol is started.
Failed	Successful	Only one authentication has been successful, limited access based on whether device or user has been authentication.

8 References

8.1 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", RFC 2119, March 1997.

[4] IEEE Standard for Local and metropolitan area networks, *Secure Device Identity*, 2009

[5] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., Wright, T., *Transport Layer Security (TLS) Extensions*, RFC 3546, June 2003

[6] Dierks, T., Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, August 2008

[7] Santesson, S., *TLS Handshake Message for Supplemental Data*, RFC 4680, September 2006

[8] Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M., *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, RFC 3820, June 2004

[9] Cooper, D., Santesson, S., Farrell, S., Boegen, S., Housley, R., Polk, W. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List Profile", RFC 5280 May 2008

[10] S.Chokhani, W.Ford, R.Sabett, C.Merrill, S.Wu, *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, RFC 3647, November 2003

[11] Polk, W., Housley, R., Bassham, L., *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3279, April 2002

[12] Schaad, J., Myers, M., *Certificate Management over CMS (CMC): Transport Protocols*, RFC 5273, June 2008

8.2 Informative References

[13] Trusted Computing Group, *TPM Main Part 2 TPM Structures*, Specification Version

1.2, March 2011

[14] Trusted Computing Group, TCG Trusted Network Connect TNC IF-T: Protocol Bindings for Tunneled EAP Methods, Specification Version 1.1, May 2007

[15] Trusted Computing Group, TCG Trusted Network Connect TNC IF-T: Binding to TLS, Specification Version 1.0, May 2009

[16] Harkins, D., Carrel, D., Cisco Systems "The Internet Key Exchange", RFC2409, November 1998,

[17] Kauffman, C., et al "Internet Key Exchange Protocol Version 2 (IKEv2), RFC 5996, September 2010,

[18] Melnikov, A., Zeilenga, K. *Simple Authentication and Security Layer (SASL)*, RFC 4422, June 2006

[19] Josefsson S., Latze C., *SASL Mechanism Family for External Authentication: EXTERNAL-**. IETF Internet Draft, Work in Progress, June 2012

[20] Rescorla, E., Ray, M., Dispensa, S., Oskov, N., *Transport Layer Security (TLS) Renegotiation Indication Extension*, RFC 5746, February 2010

[21] Trusted Computing Group, *TCG Infrastructure Working Group A CMC Profile for AIK Certificate Enrollment, Version 1.0, Revision 7*, http://www.trustedcomputinggroup.org/resources/tcg_infrastructure_working_group_a_cmc_profile_for_aik_certificate_enrollment, September 2011

[22] Nystrom, M., Kaliski, B, *PKCS#10: Certification Request Syntax Specification Version 1.7*, RFC 2986, November 2000

[23] Altman, J., Williams, N., Zhu, L., *Channel Bindings for TLS*, RFC 5929, July 2010

[24] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C., *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*, RFC 2560, June 1999

[25] Simon, D., Aboba, B., Hurst, R., *The EAP-TLS Authentication Protocols*, RFC 5216, March 2008

[26] Cam-Winget, N., McGrew, D., Salowey, J., Zhou, H., *The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)*, RFC 4851, May 2007

[27] Microsoft Corporation, [MS-PEAP]: Protected Extensible Authentication Protocol (PEAP) Specification, August 2009, available at http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS- PEAP%5D.pdf

[28] Sangster, P., Cam-Winget, N., Salowey, J., *PT-TLS: A TCP-based Posture Transport (PT) Protocol*, IETF Internet Draft, Work in Progress, August 2012

[29] Jonsson, J., Kaliski, B., *Public-Key Crypography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, RFC 3447, February 2003

[30] Zhou, H., Cam-Winget, N., Salowey, J., Hanna, S., *Tunnel EAP Method (TEAP) Version 1*, IETF Internet Draft, Work in Progress, June 2012

[31] Hoeper, K., Hanna, S., Zhou, H., Salowey, J., *Requirements for a Tunnel-Based Extensible Authentication Protocol (EAP) Method*, RFC 6678, July 2012

[32] Cam-Winget, N., Sangster, P., *PT-EAP: Posture Transport (PT) Protocol For EAP Tunnel Methods*, IETF Internet Draft, Work in Progress, July 2012

[33] Neuman, C., Yu, T., Hartman, S., Raeburn, K., *The Kerberos Network Authentication Service (V5)*, RFC 4120, July 2005

[34] Hartman, S., Zhu, L., *A Generalized Framework for Kerberos Pre-Authentication*, RFC 6113, April 2011

[35] Zhu, L., Tung, B., *Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)*, RFC 4556, June 2006

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

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

[38] The White House, National Strategy for Trusted Identities in Cyberspace, Draft, June 2010

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

[40] Congdon, Paul and Yuen, Jeffrey "Implementing IEEE 802.1AR Secure Device Identity with the Trusted Computing Group's TPM", dated 2007

[41] Trusted Computing Group, *TNC IF-MAP Metadata for Network Security*, Versions 1.1, Revision 8, http://www.trustedcomputinggroup.org/resources/tnc_ifmap_metadata_for_network_security, May 2012

[42] Salowey, J., Hanna, S., *The Network Endpoint Assessment (NEA) Asokan Attack Analysis*, RFC 6813, December 2012

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

[44] Trusted Computing Group, *TPM Main Part 3 Commands*, Specification Version 1.2, March 2011

[45] Schaad, J., Myers, M., *Certificate Management over CMS (CMC)*, RFC 5272, June 2008

[46] Trusted Computing Group, *TCG Software Stack (TSS) Specification Version 1.2 Level 1*, http://www.trustedcomputinggroup.org/resources/tcg_software_stack_tss_specification, January 2006

[47] Trusted Computing Group, *Infrastructure CMC Profile for EK/Platform Certificate Enrollment for TPMv1.2 Specification*, http://www.trustedcomputinggroup.org/resources/infrastructure_cmc_profile_for_ekplatform_certificate_enrollment_for_tpmv12_specification, April 2013

[48] Trusted Computing Group, TPM Main Part 2 Structures of the TPM, Specification Version 1.2, March 2011

[49] Sigrid, G., Carsten, R., Dirk, S., Marion, A. and Rainer, P.: *Security Evaluation of Scenarios Based on the TCG's TPM Specification.* In: Joachim, B. and Javier, L. (eds) ESORICS 2007. LNCS, vol. 4734, pp. 438

[50] Housley, R., *Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages*, RFC 4108, August 2005

9 Acronyms

For the purposes of this document, the acronyms given in Parts 2 and 3 and the following apply.

ACA	Attestation CA, formerly known as Privacy CA
AIK	Attestation Identity Key, a TPM key type
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BIOS	Basic Input/ Output System
BYOD	Bring Your Own Device
CA	Certificate Authority
CP	Certificate Policy
CPS	Certificate Practice Statement
CRMF	Certificate Request Message Format
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DNS	Domain Name System
DOS	Denial Of Service
EAP	Extensible Authentication Protocol
EAP-FAST	Extensible Authentication Protocol - Flexible Authentication via Secure Tunneling
EAP-TTLS	Extensible Authentication Protocol - Tunneled Transport Layer Security
ECDSA	Elliptic Curve Digital Signature Algorithm
EMU	EAP Method Update, an IETF work group
FAST	Flexible Authentication Secure Tunneling, a Kerberos pre-authentication protocol
HMAC	Hashed Message Authentication Code
IETF	Internet Engineering Task-Force
IKE	Internet Key Exchange
IPSec	Internet Protocol Security
IPSec AH	Internet Protocol Security Authentication Header
IPSec ESP	Internet Protocol Security Encapsulating Security Payload
ISP	Internet Service Provider
KDC	Key Distribution Center, a Kerberos component

MAP	Metadata Access Point, a TCG information security management protocol
MITM	Man-In-The-Middle
NEA	Network Endpoint Assessment, an IETF work group
OEM	Original Equipment Manufacturer
OID	Object IDentifier
PCR	Platform Configuration Register, a TPM component
PICS	Protocol Implementation Compliance Statement
PKCS	Public Key Cryptographic Standard
PKI	Public Key Infrastructure
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos
POP	Proof-Of-Possession
PWLAN	Public Wireless LAN
RADIUS	Remote Authentication Dial In User Service
RFC	Request For Comment
RFID	Radio Frequency IDentification
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman public key cryptosystem
SASL	Simple Authentication and Security Layer
SKAE	Subject Key Attestation Evidence
SRK	Storage Root Key, a TPM key type
SSL	Secure Sockets layer
TBB	Trusted Building Block
TCB	Trusted Compute Base
TGS	Kerberos Ticket Granting Service
TGT	Kerberos Ticket Granting Ticket
TLS	Transport Layer Security
TNC	Trusted Network Connect, a TCG work group
TPM	Trusted Platform Module
TSS	Trusted Software Stack
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VPN	Virtual Private Network