802.1AR™

IEEE Standard for

Local and metropolitan area networks—

Secure Device Identity

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

22 December 2009

IEEE Std 802.1AR™-2009

IEEE Standard for
Local and metropolitan area networks—

Secure Device Identity

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 9 December 2009

IEEE SA-Standards Board

Abstract: A secure device identifier (DevID) is cryptographically bound to a device and supports authentication of the device's identity. Locally significant identities can be securely associated with an initial manufacturer-provisioned DevID and used in provisioning and authentication protocols to allow a network administrator to establish the trustworthiness of a device and select appropriate policies for transmission and reception of data and control protocols to and from the device.
Keywords: access control, authentication, authorization, certificate, LANs, local area networks, MAC security, MANs, metropolitan area networks, PKI, port-based network access control, secure association, secure device identifier, security, X.509

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-7381-6141-9 STD96035
Print: ISBN 978-0-7381-6142-6 STDPD96035

Introduction

This introduction is not part of IEEE Std 802.1AR-2009, IEEE Standard for Local and metropolitan area networks—Secure Device Identity.

A secure device identifier (DevID) is a cryptographic identity bound to a device used for assertion of the device's identity. IEEE Std 802.1AR specifies

— 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.

IEEE Std 802.1AR can be used in conjunction with IEEE Std 802.1X™ [B2] and other IEEE and industry standards that require a secure identifier or credential as part of authentication and provisioning processes that establish trust in a device.[1]

This is the first edition of IEEE Std 802.1AR.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association website at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

[1]The numbers in brackets correspond to those of the bibliography in Annex D.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at http://standards.ieee.org.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses. Other Essential Patent Claims may exist for which a statement of assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the IEEE 802.1 Working Group had the following membership:

Tony Jeffree, *Chair*
Paul Congdon, *Vice Chair*
Mick Seaman, *Chair, Security Task Group*
Max Pritikin, *Editor*

Zehavit Alon
Jan Bialkowski
Rob Boatright
Jean-Michel Bonnamy
Paul Bottorff
Rudolf Brandner
Craig W. Carlson
Weiying Cheng
Rao Cherukuri
Jin-Seek Choi
Don O. Connor
Diego Crupnicoff
Claudio Desanti
Zhemin Ding
Linda Dunbar
David Elie-Dit-Cosaque
Janos Farkas
Donald Fedyk
Norman Finn
Robert Frazier
John Fuller
Geoffrey Garner
Anoop Ghanwani
Franz Goetz
Yannick Goff
Eric Gray
Karanvir Grewal
Craig Gunther
Mitch Gusat
Stephen Haddock
Asif Hazarika
Charles Hudson
Romain Insler
Pankaj K. Jha
Abhay Karandikar
Prakash Kashyap
Hal Keen
Keti Kilcrease
Doyeon Kim
Yongbum Kim
Philippe Klein
Mike Ko
Vinod Kumar
Bruce Kwan
Kari Laihonen
Ashvin Lakshmikantha
John Lemon
Michael Lerer
Marina Lipshteyn
Gael Mace
Ben Mack-Crane
David Martin
Alan McGuire
James McIntosh
Menucher Menuchery
John Messenger
Gabriel Montenegro
Matthew Mora
John Morris
Eric Multanen
Paul Nikolich
Kevin Nolish
David Olsen
Donald Pannell
Glenn Parsons
Joseph Pelissier
David Peterson
Hayim Porat
Karen T. Randall
Josef Roese
Derek J. Rohde
Dan Romascanu
Jessy V. Rouyer
Jonathan Sadler
Ali Sajassi
Joseph Salowey
Panagiotis Saltsidis
Satish Sathe
John Sauer
Koichiro Seto
Himanshu Shah
Nurit Sprecher
Kevin B. Stanton
Robert A. Sultan
Muneyoshi Suzuki
Michael J. Teener
Patricia Thaler
Oliver Thorp
Manoj Wadekar
Yuehua Wei
Brian Weis
Martin White
Bert Wijnen
Michael D. Wright
Chien-Hsien Wu
Ken Young
Glen Zorn

The following members of the balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Thomas Alexander
Butch Anton
Hugh Barrass
Tomo Bogataj
William Byrd
James Carlo
Juan Carreon
Peng Chen
Keith Chow
Charles Cook
Russell Dietz
Thomas Dineen
Sourav Dutta
Donald Eastlake, 3rd
C. Fitzgerald
Yukihiro Fujimoto
Venkatasubramaniyan Ganesan
Devon Gayle
Michael Geipel
Randall Groves
C. Guy
Joseph Gwinn
John Harauz
John Hawkins
Marco Hernandez
Atsushi Ito
Raj Jain
Tony Jeffree
Shinkyo Kaku
Piotr Karocki
Stuart J. Kerry
Chad Kiger
Yongbum Kim
Shen Loh
William Lumpkins
G. Luri
Jonathon McLendon
Gary Michel
Michael Montemurro
Jose Morales
Andrew Myles
Michael S. Newman
Nick S. A. Nikjoo
Satoshi Obara
Hayim Porat
Max Pritikin
Karen T. Randall
Robert Robinson
Jessy V. Rouyer
Randall Safier
Joseph Salowey
John Sargent
Bartien Sayogo
Mick Seaman
Gil Shultz
Christopher Sonnek
Kapil Sood
Amjad Soomro
Thomas Starai
Rene Struik
Walter Struppler
Robert A. Sultan
Joseph Tardo
William Taylor
Patricia Thaler
Mark-Rene Uchida
Dmitri Varsanofiev
Prabodh Varshney
Brian Weis
Michael D. Wright
Kunpeng Wu
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 9 December 2009, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

John Barr
Karen Bartleson
Victor Berman
Ted Burse
Richard DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Ted Olsen
Glenn Parsons
Ronald C. Petersen
Narayanan Ramachandran
Jon Walter Rosdahl
Sam Sciacca

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Howard L. Wolfman, *TAB Representative*
Satish K. Aggarwal, *NRC Representative*
Michael Janezic, *NIST Representative*

Lorraine Patsco
IEEE Standards Program Manager, Document Development

Michael D. Kipness
IEEE Standards Program Manager, Technical Program Development

Contents

IEEE Standard for
Local and metropolitan area networks—

Secure Device Identity

1. Overview

IEEE 802® Local Area Networks (LANs) are often deployed in networks that provide publicly accessible service access or that cannot be completely physically secured. The protocols that configure, manage, and regulate access to these networks and network-based services and applications typically run over the networks themselves. Secure and predictable operation of such networks depends on authenticating each device attached to and participating in the network, so that the degree of trust and authorization to be accorded to that device by its communicating peers can be determined.

Authentication of a human user, through a credential known to or possessed by that user, is often used to authenticate users of devices such as laptop personal computers. However many of the devices that compose a network are designed for unattended autonomous operation and might not support user authentication. These include the routers and bridges that interconnect and provide access to the LANs. Further, the previously common assumption that network access controls were to provide protection of the network against abuse through unauthenticated and unauthorized access, while offering no protection to the accessing devices, is now known not only to expose those devices but also the network itself. Failure to provide devices that access the network with the mutual guarantee that they are connected to legitimate network access points allows malicious devices to interpose themselves between the network and its authenticated and authorized users, and effectively make use of the credentials of the latter. For these reasons a secure device identifier, i.e., one that embodies an authentication credential that cannot be easily removed or copied for use in a device under the control of someone who wishes to gain unauthorized access to or attack the operation of a network, is highly desirable.

Protocols for configuring, managing and regulating access to a network depend on the existence of a device identifier or human authentication of initial access to associate a device with an authentication credential.

This results in a “chicken-and-egg” scenario, wherein these credentials often must be installed during an expensive “pre-provisioning” process before actual deployment. Even when device credentials are deployed in-place, the process is often interactive, involving a physically secured connection to the device being deployed and a knowledgeable system administrator.

Secure Device Identifiers (DevIDs) are designed to be used as interoperable secure device authentication credentials with Extensible Authentication Protocol (EAP) and other industry standard authentication and provisioning protocols. A standardized device identity facilitates interoperable secure device authentication that helps simplify and standardize secure deployment and management of devices. This standard will be of benefit to manufacturers of conformant LAN equipment, their customers, and users of LANs or LAN services that are based on such equipment.

A device with DevID capability incorporates a globally unique manufacturer provided Initial Device Identifier (IDevID), stored in a way that protects it from modification. The device may support the creation of Locally Significant Device Identifiers (LDevIDs) by a network administrator. Each LDevID is bound to the device in a way that makes it infeasible for it to be forged or transferred to a device with a different IDevID without knowledge of the private key used to effect the cryptographic binding. LDevIDs can incorporate, and fully protect, additional information specified by the network administrator to support local authorization conventions. LDevIDs may also be used as the sole identifier (by disabling the IDevID) to assure the privacy of the user of a DevID and the equipment in which it is installed.

1.1 Scope

This standard specifies unique per-device identifiers (DevID) 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.

1.2 Purpose

This standard defines a standard identifier for IEEE 802 devices that is cryptographically bound to that device, and defines a standard mechanism to authenticate a device’s identity. A verifiable unique device identity allows establishment of the trustworthiness of devices. This facilitates secure device provisioning.

1.3 Relationship to other standards

The present work has been undertaken to provide an identifier that is generally useful across IEEE 802 networks. It draws on and is informed by other standards that have been developed elsewhere for different purposes. Where possible, this work attempts compatibility with the following standards:

a) Trusted Platform Module (TPM)

b) Worldwide Interoperability for Microwave Access (WiMAX)

c) Extensible Authentication Protocol-Transport Layer Security (EAP-TLS)

For other standards that have influenced the development of this standard or are of general interest see the bibliography (Annex D).

2. Normative references

The following referenced documents are indispensable for the application of this standard (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).[2]

IETF RFC 2578, STD 58, Structure of Management Information for Version 2 (SMIv2), McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., Waldbusser, S., April 1999.[3]

IETF RFC 2579, STD 58, Textual Conventions for SMIv2, McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., Waldbusser, S., April 1999.

IETF RFC 2580, STD 58, Conformance Statements for SMIv2, McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., Waldbusser, S., April 1999.

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

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

IETF RFC 3410, Introduction and Applicability Statements for Internet Standard Management Framework, Case, J., Mundy, R., Partain, D., Stewart, B., December 2002.

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

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

IETF RFC 4086, Randomness Requirements for Security, Eastlake 3rd, D., Schiller, J., Crocker, S., June 2005.

IETF RFC 4133, Entity MIB (Version 3), Bierman, A., McCloghrie, K., August 2005.

IETF RFC 4492, Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS), Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., Moeller, B., May 2006.

IETF RFC 4949, Internet Security Glossary, Version 2, Shirey, R., August 2007.

IETF RFC 5008, Suite B in Secure/Multipurpose Internet Mail Extensions (S/MIME), Housley, R., Solinas, J., September 2007.

IETF RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W., May 2008.

[2]ANSI publications are available from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[3]IETF RFCs are available from the Internet Engineering Task Force Web site at http://www.ietf.org/rfc.html.

IETF RFC 5289, TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM), Rescorla, E., August 2008.

IETF RFC 5480, Elliptic Curve Cryptography Subject Public Key Information, Turner, S., Brown, D., Yiu, K., Housley, R., Polk, T., March 2009

ISO/IEC 15802-1:1995, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 1: Medium Access Control (MAC) service definition.[4]

NIST FIPS 140-2, Annex C: Approved Random Number Generators.[5]

NIST FIPS 180-3, Secure Hash Standard (SHS), October 2008.

NIST FIPS 186-3, Digital Signature Standards (DSS), June 2009.

Standards for Efficient Cryptography (SEC), "SEC 1: Elliptic Curve Cryptography," Certicom Research, 2000.[6]

[4]ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[5]NIST publications are available from the National Institute of Standards and Technology, NIST Public Inquiries, NIST, 100 Bureau Drive, Stop 3460, Gaithersburg, MD, 20899-3460, USA (www.nist.gov).

[6]This document is available at http://www.secg.org/collateral/sec1_final.pdf.

3. Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms & Definitions* should be referenced for terms not defined in this clause.[7]

aggregate device: A device containing multiple logical or physical devices.

authentication: The process of verifying an identity claimed by or for a system entity. Defined in RFC 4949.[8]

authentication exchange: The two-party conversation between systems performing an authentication process.

NOTE—Examples of authentication exchange methods are Extensible Authentication Protocol (EAP) and Simple Authentication and Security Layer (SASL).[9]

authentication process: The cryptographic operations and supporting data frames that perform the actual authentication.

Authenticator: An entity that facilitates authentication of other entities attached to the same LAN.

binding: *See:* **cryptographic binding.**

certificate revocation list (CRL): A signed list of revoked certificates. Defined in RFC 5280.

certificate signing request (CSR): A signed message from the device to the Certification Authority (CA) requesting a certificate be issued. This is a variant of the RFC 2986 defined CertificationRequest with additional specifications to enable the use of Elliptic Curve Cryptography (ECC).

cipher suite: A set of one or more algorithms, designed to provide any number of the following: data confidentiality, data authenticity, data integrity, replay protection.

client: The protocol entity that makes use of a service.

credential: A data object that provides authentication information such as a public key along with other identity fields. The entity identified by a credential normally possesses a secret or private key that can be used to prove its identity.

cryptographic binding: A data object constructed using cryptographic operations to combine a secret with other arbitrary data objects such that it may be proven that the result could only be created by an entity having knowledge of the secret.

cryptographic key: A parameter that determines the operation of a cryptographic function such as the following:

a) The transformation from plain text to cipher text and vice versa
b) Synchronized generation of keying material
c) Digital signature computation or validation[10]

[7] *The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.

[8] Information on references can be found in Clause 2.

[9] Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

[10] This and other definitions in this clause have been drawn from ASC X9 TR1, Technical Report for ABA ASC/X9 Standards Definitions, Acronyms, and Symbols, 2002.

device: A device is any entity that has a single IDevID credential.

IEEE 802 Local Area Network (LAN): IEEE 802 LANs are LAN technologies that provide a MAC Service equivalent to the MAC Service defined in ISO/IEC 15802-1:1995. IEEE 802 LANs include IEEE 802.3™ (CSMA/CD), IEEE 802.11™ (wireless), and IEEE 802.17™ (resilient packet ring).

NOTE—In this standard, also referred to simply as *LANs*.

Initial Secure Device Identifier (IDevID): The Secure Device Identifier installed on the device by the manufacturer.

key: *See:* **cryptographic key.**

Locally Significant Secure Device Identifiers (LDevIDs): A Secure Device Identifier credential that is unique in the local administrative domain in which the device is used.

Secure Device Identifier (DevID): A device identifier that is cryptographically bound to the device and is composed of the Secure Device Identifier Secret and the Secure Device Identifier Credential.

Secure Device Identifier Credential (DevID Credential), Secure Device Identifier Certificate (DevID Certificate): Synonym for a data object constructed using cryptographic operations to bind the DevID Name and other data to the cryptographic keying material possessed by the device.

Secure Device Identifier Module (DevID Module): A logical security component that will securely store and operate on one or more DevID Secret(s) and associated DevID Credentials.

Secure Device Identifier Root Credential Store: The database of root credentials for IDevID and LDevID credentials that are stored and used by a DevID compliant solution. This is equivalent to the common Web browser root store and may be shipped with the solution.

Secure Device Identifier Secret (DevID Secret): The private key portion of a public-private key pair bound to a DevID Credential.

zeroization: A method of erasing electronically stored data, cryptographic keys, and critical stored parameters by altering or deleting the contents of the data storage to prevent recovery of the data.

NOTE—Adapted from NIST Special Publication 800-57 [B32].[11]

[11]The numbers in brackets correspond to those of the bibliography in Annex D.

4. Acronyms and abbreviations

CA	Certification Authority

NOTE—CA is also an abbreviation for secure Connectivity Association in IEEE Std 802.1AE™ and IEEE Std 802.1X™.

CPS	Certification Practices Statement
CRL	Certificate Revocation List
CSR	Certificate Signing Request
EAP	Extensible Authentication Protocol
EAP-TLS	EAP Transport Layer Security
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
IP	Internet Protocol
LAN	IEEE 802 Local Area Network
LMI	Layer Management Interface
MAC	Media Access Control
MIB	Management Information Base
NIST	National Institute of Standards and Technology
PICS	Protocol Implementation Conformance Statement
PKCS	Public-Key Cryptography Standards
RADIUS	Remote Authentication Dial In User Service
RFC	IETF Request for Comments
RNG	random number generator
RSA	Ron Rivest, Adi Shamir, and Leonard Adleman cryptography algorithm
SASL	Simple Authentication and Security Layer
SHA	Secure Hash Algorithm
SNMP	Simple Network Management Protocol
SMI	Structure of Management Information
TLS	Transport Layer Security
TPM	Trusted Platform Module
WiMAX	Worldwide Interoperability for Microwave Access

5. Conformance

A claim of conformance to this standard is a claim that the behavior of an implementation of a Secure Device Identifier (IDevID and LDevID) meets the requirements of this standard as they apply to the format of the identifier, secure storage of it, and management of the identifier. Example uses of the DevID are provided but are not limited to the discussions in this standard.

Conformance to this standard does not ensure that a system implementing the standard is secure, nor does it ensure that the operation of protocols relying upon capabilities from this standard are immune to breaches by an attacker.

5.1 Requirements terminology

For consistency with existing IEEE standards, requirements placed upon conformant implementations of this standard are expressed using the following terminology:

a) *shall* is used for mandatory requirements
b) *may* is used to describe implementation or administrative choices (may means is permitted to, and hence, may and may not mean precisely the same thing)
c) *should* is used for recommended choices (the behaviors described by should and should not are both permissible but not equally desirable choices)

The PICS proforma (see Annex A) reflects the occurrences of the words shall, may, and should within the standard.

The standard avoids needless repetition and apparent duplication of its formal requirements by using *is*, *is not*, *are*, and *are not* for definitions and the logical consequences of conformant behavior. Behavior that is permitted but is neither always required nor directly controlled by an implementer or administrator, or whose conformance requirement is detailed elsewhere, is described by *can*. Behavior that never occurs in a conformant implementation or system of conformant implementations is described by *can not*. The word *allow* is used as a replacement for the cliche “support the ability to...,” and the word *capability* means “is configurable such that....”

5.2 Protocol Implementation Conformance Statement

The supplier of an implementation that is claimed to conform to this standard shall complete a copy of the PICS proforma provided in Annex A and shall provide the information necessary to identify both the supplier and the implementation.

5.3 Required capabilities

A device for which conformance to this standard is claimed shall:

a) Contain an IDevID as specified in 6.2.1 (DevID credentials).
b) Contain the credential chain as specified in 6.2.1 (DevID credentials).
c) Support an RSA or ECDSA key as specified in 6.2.2 (Asymmetric cryptography).
d) Generate random numbers as specified in 6.2.4 (Random number generator).
e) Store an IDevID secret as specified in 6.2.5 (Storage).
f) Support the mandatory operations as specified in 6.3 (DevID Service Interface).
g) Support the DevID credential format as specified in Clause 7 (DevID Credential details).

A device supporting Simple Network Management Protocol (SNMP) to manage the capabilities of DevIDs for which conformance to this standard is claimed shall:

h) Restrict access to SNMP version 3 and higher, and only in privacy/authentication mode as specified in 6.3.12 (LDevID Credential insert).

i) Implement the MIB Management Interface as specified in 6.4 (DevID Management Interface).

5.4 Optional capabilities

A device for which conformance to this standard is claimed may:

a) Support LDevID credentials as specified in 6.2 (Components of a DevID module).

b) Store DevID secrets externally to the DevID module as specified in 6.2.5 (Storage).

c) Support the optional operations specified in 6.3 (DevID Service Interface).

d) Share an LDevID key pair among multiple LDevID credentials as specified in 6.3.12 (LDevID Credential insert).

e) Externally generate the IDevID secret as specified in 6.5.1 (Creation of DevID secret).

f) Share the IDevID secret with LDevID credentials as specified in 6.5.1 (Creation of DevID secret).

5.5 Recommended capabilities

A device for which conformance to this standard is claimed should:

a) Include Public Key Infrastructure (PKI) hierarchy root credentials in an easily discovered location as specified in 6.2.1 (DevID credentials).

b) Support at least one LDevID credential as specified in 6.2.1 (DevID credentials).

c) Encrypt externally stored secrets as specified in 6.2.5 (Storage).

d) Support the recommended operations as specified in 6.3 (DevID Service Interface).

e) Confirm that inserted LDevID credentials match this standards credential profile for LDevID credentials 6.3.12 (LDevID Credential insert).

f) Publish the Certification Authority (CA) Certification Practices Statement (CPS) in the PICS as specified in 6.5.1 (Creation of DevID secret).

g) Generate an independent LDevID secret as specified in 6.5.1 (Creation of DevID secret).

h) Generate IDevID secret internally as specified in 6.5.1 (Creation of DevID secret).

i) Report key generation mechanism in CPS as specified in 6.5.1 (Creation of DevID secret).

j) Document key exposure plans in their PICS as specified in 6.6.1 (Mitigating risks of key exposure).

k) Support identity hiding as specified in 6.6.3 (Identity Hiding).

The supplier for a device for which conformance to this standard is claimed should:

l) Have a unique issuer name in the IDevID as specified in 7.2.4 (issuer).

6. Secure Device Identifier Module

This clause describes the logical components of the DevID Module. It provides the necessary context for understanding the security requirements (and acceptable practical limitations on those requirements) for the creation, use, maintenance, and deletion of DevID credentials. An overview of DevID credentials is provided although the details of DevID fields are specified in Clause 7 (DevID Credential details).

Informative use cases are provided in Annex C as examples.

6.1 What is a device?

A device is any entity in an IEEE 802 LAN that seeks to obtain services from the network or provide services on the network. A device is defined by having a single IDevID credential.

Multiple logical or physical devices may be contained within an aggregate device. Each of these logical or physical devices will have its own unique DevID. The DevID used to identify the aggregate device may depend on the context under which the identification of the aggregate device is done. The selection of a DevID based on this context is beyond the scope of this document. Service interfaces defined in 6.3 do not refer to aggregate devices.

The DevID module is a logical security component that stores and operates on the DevID secrets and associated DevID credentials. The PICS shall describe the device the DevID module is identifying.

6.2 Components of a DevID module

The components of a DevID module are shown in Figure 6-1. Implementations may provide additional functionalities. The DevID credential and DevID secret are used by the DevID module to prove the identity of the device. It is imperative that the DevID secret is securely protected from access by any other entity. The DevID credential is publicly available to any entity that wishes to verify the identity of a device.

6.2.1 DevID credentials

The DevID module shall include the IDevID and zero or more LDevIDs. At least one LDevID credential should be supported.

The DevID module shall include the entire credential chain up to the root credential and should include the root credential. For a shallow hierarchy this requirement could be met with only the IDevID credential on the device. The root certificates should be shipped along with the device or made available by the solution manufacturer in a reasonably discoverable fashion such as on the manufacturer's Web site.

The DevID credential format shall be as described in Clause 7 (DevID Credential details).

DevIDs have to be explicitly created. IDevIDs are normally created during manufacturing or initial provisioning, and can be generated internally in the DevID module or can be created externally and subsequently transferred to secure storage within the DevID module. LDevIDs can be created at any time, in accordance with local policies, and likewise can be created internally in the DevID module, or can be created externally and transferred to the DevID module.

Following creation, DevIDs enter their active use lifetime, during which they can be used and maintained in accordance with the administrative policies of the network and device owners possessing them. IDevIDs remain in their active use state until the effective lifetime ends or the IDevID is disabled. LDevIDs can be managed by the operations detailed in 6.3 (DevID Service Interface).

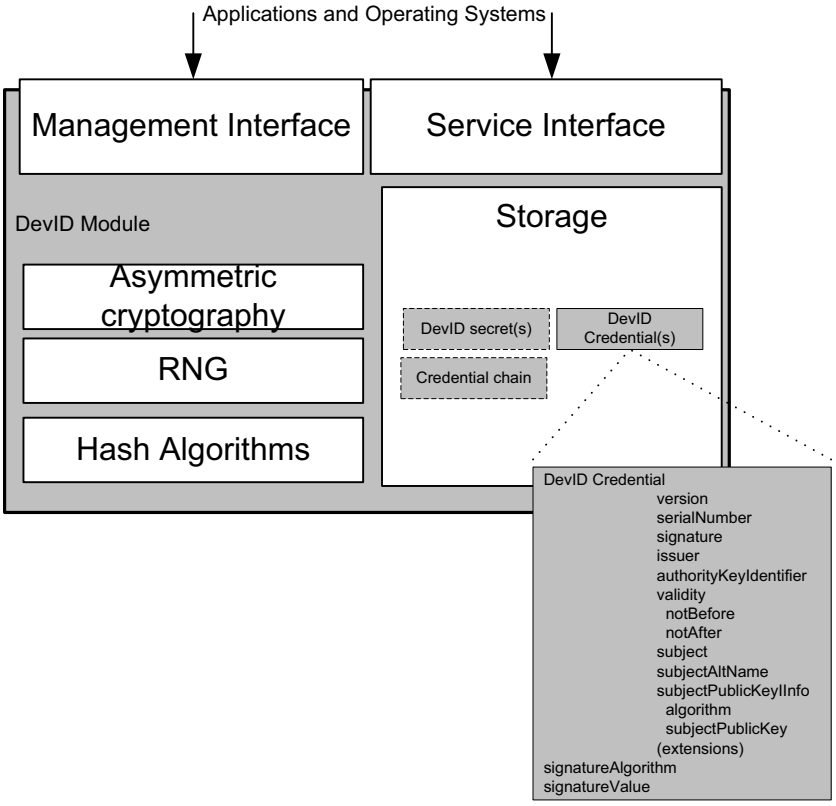

Figure 6-1— DevID components

A DevID (particularly an IDevID) continues 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. The GeneralizedTime value “99991231235959Z” is used for this purpose.

The CA certificates of the certificate chain shall conform to the IDevID specification as specified in 7.1 (DevID hierarchy credential fields).

6.2.2 Asymmetric cryptography

The DevID secret shall be either the secret portion of a RSA private-public key pair as defined in 7.3.1 (RSASSA-PKCS1-v1.5) or the secret portion of an P-256 elliptic curve private-public key pair as defined in 7.3.2 (ECDSA).

Since the DevID secret is used only within the DevID module in which it is installed, the underlying representation is a choice left to the implementer. Possible representations are discussed in 6.2.5 (Storage) along with estimates of the storage requirements.

Techniques for finding a suitable DevID secret are discussed in IEEE Std 1363™-2000 [B4] and NIST FIPS 186-3.

6.2.3 Hash algorithms

The DevID Module shall contain an implementation of the hash algorithm as specified in 7.3.3 (SHA-256 hash algorithm).

6.2.4 Random number generator

The system shall provide a strong random number generator (RNG) for the generation of all numbers described as random in this standard. If a non-deterministic RNG (e.g., hardware RNG) is not available, the system shall make use of sufficient entropy to create a good quality seed for a deterministic RNG and which should be one of those listed in FIPS 140-2, Annex C: Approved Random Number Generators. Additionally RFC 4086 provides a self-described “best current practice” document for producing random numbers.

6.2.5 Storage

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.

In practice, there are different ways in which the DevID secret can be stored. The most direct is to provide hardware secured storage within the module for the IDevID secret (e.g., one-time programmable memory embedded within the DevID module). Other implementation possibilities exist such as a nominally unsecured storage location where the device's operating system restricts all access (software secured storage).

When using an external storage mechanism, there will be additional overhead for the storage of encryption keys; for example if AES 128-bit encryption is used, the 16-octet AES key must be stored within the DevID module. If external storage is used, the protection method (possibly encryption) shall be documented in the PICS.

Example storage sizes for the DevID secret, as maintained in various formats, are listed in Table 6-1.

Table 6-1—DevID storage examples

Name	Type	Length (octets)	Value
RSAPrivateKey	The portable PKCS1 format, including public and private exponents and all associated information. This is not size optimized.	~1188	The following are included in an RSAPrivateKey: version modulus (INTEGER, n) publicExponent (INTEGER, e) privateExponent (INTEGER, d) prime1 (INTEGER, p) prime2 (INTEGER, q) exponent1 (INTEGER, d mod (p–1)) exponent2 (INTEGER, d mod (q–1)) coefficient (INTEGER ((inverse of q) mod p)
RSA modulus (n) and private exponent (d)	Bits.	~(257 + 256)	The computed modulus (n = pq) and private exponent (d).
ECPrivateKey	The Standards for Efficient Cryptography Group format for an elliptic curve private key. This is for a prime256v1 key.	~119	The following are included in an ECPrivateKey: version privateKey (OCTET STRING) parameters publicKey (BIT STRING)
Elliptic Curve	Bits	32	Just the privateKey

6.3 DevID Service Interface

The DevID module operational functionalities provide direct control over the device's identity. Access to these functionalities should be limited to the device administrator and owner as is appropriate for the device and administration model. The details of the security considerations and resulting implementations are out of the scope of this standard. Appropriate techniques might include *role-based access control, user level access, administrative level access,* etc.

Operations included in 6.3 are core functionalities that make the DevID module useful for assertion of the DevID identity. Examples of these operations in use are found in Annex C. Explicit encryption and decryption operations are not included.

The DevID module shall implement a service interface that supports the following operations:

— Initialization
— Enumeration of the DevID public keys
— Enumeration of DevID credentials
— Enumeration of DevID credential chain
— Signing
— Enable/disable DevID credential
— Enable/disable of DevID key

The DevID module should implement the following operations:

— LDevID Key generation
— LDevID Key insertion
— LDevID Key delete
— LDevID Certificate Signing Request generation
— LDevID Credential insert
— LDevID Credential Chain insert
— LDevID Credential delete
— LDevID Credential Chain delete

The DevID module may implement the following operations:

— Addition of RNG entropy

The enumerated types shown in Table 6-2 are defined for the service interface.

Table 6-2—Enumerated key types

KEYTYPE_RSA	The format of the key data shall be an RFC 3447 defined RSAPrivateKey.
KEYTYPE_ECC	The format of the key data shall be an OCTET STRING of the ECDSA private key. The FIPS recommended curve prime256v1 shall be supported.

NOTE—Insertion of replacement IDevID credentials is to be limited to the manufacturer operations and is out of scope of this standard. Replacing the IDevID is tantamount to destroying the current device and replacing it with a new device with the new IDevID credential.

6.3.1 Initialization

The DevID module shall implement an initialization operation. When called this operation causes the DevID module to prepare itself for use. For example any self-test operations necessary to comply with FIPS 140-2 [B31] would be performed at this time.

This operation is not expected to be exposed to the device administrator but rather to be used by the OS or firmware to properly initialize the DevID module. The DevID module returns FALSE if its internal consistency checks fail, if it detects a compromise of its protected data, or if it has been instructed to disable access by an authorized management interface.

Inputs: None
Output: A binary value indicating availability (TRUE) or unavailability (FALSE) of the other DevID Service Interface operations.

6.3.2 Enumeration of DevID public keys

The DevID module shall implement this operation. This operation allows the administrator to determine the DevID keys installed in the DevID module.

All keys are indexed internally with an integer within [0..n–1], and the index of the key is returned as the "keyIndex." Any negative value of keyIndex is indicative of an internal error. The IDevID key shall always be at index "0." Operations on keys use the keyIndex to address a specific key.

The key material returned shall be limited to the public key material.

Inputs: None
Output: A table containing, for each key, the key's keyIndex, a value indicating if the key is enabled and the public key material itself.
This operation may be available via a remote management protocol.

6.3.3 Enumeration of DevID credentials

The DevID module shall implement this operation. This operation allows the administrator to determine the DevID credentials installed in the DevID module.

All credentials are indexed internally with an integer within [0..n–1], and the index of a credential is returned as the "credentialIndex." Any negative value of credentialIndex is indicative of an internal error. The IDevID credential shall always be at index "0." Additional operations on credentials use the credentialIndex to address a specific credential.

Inputs: None
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.
This operation may be available via a remote management protocol.

NOTE—There is no correspondence between the keyIndex and credentialIndex other than that "0" indexes the IDevID elements.

6.3.4 Enumeration of a DevID credential chain

The DevID module shall implement this operation. This operation allows the administrator to enumerate a credential chain associated with a specified DevID credential. The root certificate may be part of the chain.

Inputs: credentialIndex of the DevID credential for which the credential chain is requested
Output: A table containing each credential within the credential chain.
This operation may be available via a remote management protocol.

6.3.5 Signing

The DevID module shall implement this operation.

This operation takes the DevID keyIndex on input so that the signing operation can be specified prior to the availability of a DevID credential. This is useful when using the signing operation during enrollment operations.

The currentEncoding specifies the current encoding of the data as specified in 6.3.5.1 (RSA Signing) and 6.3.5.2 (ECDSA Signing).

Inputs: keyIndex, currentEncoding, dataLength, dataOctets. enum currentEncoding_t {PKCS1HASH_SHA256, PKCS1DIGESTINFO_OPAQUE, ECDSADIGESTINFO_OPAQUE}
Output: A value indicating success or failure of the operation, the signature.
This operation shall not be available via a remote management protocol.

6.3.5.1 RSA Signing

If the key is an RSA key, then this operation shall generate a PKCS1 signature as defined in RFC 3447 with the signature algorithm of RSASSA-PKCS1-v1_5, for maximum interoperability.

The currentEncoding specifies the current encoding of the data. The dataOctets are partially encoded for RFC 3447 signing prior to calling this DevID module interface. A value of PKCS1HASH_SHA256 indicates that the dataOctets are a SHA256 hash of the original data as specified by RFC 3447 id-sha256. The DevID Module will continue encoding the data, starting at RFC 3447 Section 9.2 step 2, by building and padding the DigestInfo ASN.1 value and then building the full PKCS1 signature.

A currentEncoding value of PKCS1DIGESTINFO_OPAQUE indicates that the dataOctets are already encoded into the equivalent of the RFC 3447 Section 9.2 step 2 specified DigestInfo. The DevID Module will continue encoding the data, starting at RFC 3447 Section 9.2 step 3, by padding the dataOctet as specified for the DigestInfo ASN.1 value, and then building the full PKCS1 signature.

NOTE—Some uses of PKCS1 specify an alternative to the RFC 3447 DigestInfo structure. For example TLS 1.1 RFC 4346 specifies “a 36-byte structure of two hashes (one SHA and one MD5).” The use of PKCS1DIGESTINFO_OPAQUE provides support for this type of construct. It also provides a mechanism for components leveraging the DeviceID Module to obtain PCKS1 signatures using legacy hash algorithms such as SHA-1 or MD5.

6.3.5.2 ECDSA Signing

If the key is an ECC key then this operation shall generate an ECDSA signature as defined in ANSI X9.62-2005 with the signature algorithm as specified in RFC 4492 Section 5.10.

The currentEncoding value of ECDSADIGESTINFO_OPAQUE indicates that the dataOctets are a computed hash of the original data. In order to use hash functions stronger than SHA-1, TLS ECC cipher suites using SHA-256 and SHA-384 have been specified in RFC 5289. For HMAC-based cipher suites, Section 3.1 of RFC 5289 should be used.

6.3.6 Enable/disable DevID credential

The DevID module shall implement this operation. The enable/disable operation is used by the administrator to control the availability of DevID credentials. Disabling a specific DevID credential could have subsequent authentication or authorization ramifications as the device will no longer be able to assert its identity by using this credential. This operation allows the device administrator to prevent use of a particular credential without disabling the associated DevID key, which might be in use by a different credential.

Re-enabling a disabled DevID credential makes it fully available. The credentialIndex of a disabled DevID credential is only valid as input to this operation and the LDevID Credential delete operation (6.3.14).

Inputs: The credentialIndex of the DevID credential to be enabled or disabled. The state (enable or disable) desired.
Output: A value indicating success or failure of the operation.
This operation may be available via a remote management protocol.

6.3.7 Enable/disable DevID key

The DevID module shall implement this operation. The enable/disable operation is used by the administrator to control the availability of DevID keys. Disabling a specific DevID key could have subsequent authentication or authorization ramifications as the device will no longer be able to assert its identity by using this key. This allows the device administrator to maintain a measure of privacy by limiting exposure of the device's cryptographic identity.

A disabled DevID key is not deleted or modified in any way while it is disabled (unless it is explicitly deleted). Re-enabling a disabled DevID key makes it fully available. The keyIndex of a disabled DevID key is only valid as input to this operation and the LDevID key delete operation (6.3.10).

Inputs: The index of the DevID key to be enabled or disabled. The state (enable or disable) desired.
Output: A value indicating success or failure of the operation.
This operation may be available via a remote management protocol

NOTE—Disabling a DevID key will render the credentials unusable, and they should also be disabled.

6.3.8 LDevID Key generate

The DevID module should implement this operation. This operation allows for the administrator to generate additional LDevID key material within the DevID module.

Key material shall be disabled by default and must be explicitly enabled before use; see 6.3.7 (Enable/ disable DevID key).

Inputs: Type of key specified as either KEYTYPE_RSA or KEYTYPE_ECC. The size of the key material.
Output: A value indicating success or failure of the operation. The keyIndex of the new key material.
This operation may be available via a remote management protocol.

6.3.9 LDevID Key insert

The DevID module should implement this operation. This operation allows for the administrator to generate additional LDevID key material and then insert it into the DevID module.

Key material shall be disabled by default and must be explicitly enabled before use; see 6.3.7 (Enable/ disable DevID key).

Inputs: Type of key specified as either KEYTYPE_RSA or KEYTYPE_ECC. The key material. The size of the key material.
Output: A value indicating success or failure of the operation. The keyIndex of the new key material.
This operation may be available via a remote management protocol.

6.3.10 LDevID key delete

The DevID module should implement this operation. This operation allows the administrator to delete an LDevID keypair by specifying its keyIndex. The DevID module may perform cryptographic zeroization on LDevID key material as part of the key delete process.

After key deletion the LDevID credential is no longer useful, since the DevID module can no longer perform associated signing operations. It is expected that the initiator of this operation will also delete any associated LDevID credential.

This operation shall not affect the IDevID even if an LDevID uses the IDevID keypair. Additional mechanisms beyond these runtime operations may be defined to support deleting of the IDevID key, which is logically equivalent to decommissioning the device.

Inputs: The keyIndex
Output: A value indicating success or failure of the operation.
This operation may be available via a remote management protocol.

6.3.11 LDevID Certificate Signing Request generate

The DevID module should implement this operation. The subjectname information for the Certificate Signing Request (CSR) may come from the IDevID credential, although the device may support operations to customize this behavior and the CA infrastructure is expected to have ultimate authority.

If the key is an RSA key, then this operation shall generate a CSR using the RFC 2986 defined CertificationRequest.

If the key is an ECC key then this operation shall generate a CSR using the RFC 2986 defined CertificationRequest with the use of a subjectPublicKeyInfo and signature as specified in RFC 3279 and additionally described in 7.2.10 (subjectPublicKeyInfo) and 7.2.12 (signatureValue).[12]

Inputs: The keyIndex and the hashAlgorithm
Output: A value indicating success or failure of the operation. A CSR.
This operation may be available via a remote management protocol.

6.3.12 LDevID Credential insert

The DevID module should implement this operation. This operation allows LDevID credentials to be installed into the LDevID module. LDevID Credentials can be installed at any time. Multiple LDevID credentials may be associated with the same keypair.

There is no stateful mapping between a previously generated CSR and the resulting LDevID credential. Because of this it is possible for a network management system to generate a locally significant LDevID credential based entirely upon the IDevID credential presented by the device, allowing low-end devices to support LDevIDs without also needing to support key generation. There is no mapping between the IDevID credential fields and the LDevID credential fields.

The DevID module shall confirm that the inserted credential contains a public key that matches an existing DevID Module private key. The DevID module should confirm that the inserted credential is an LDevID as in 6.2.1 (DevID credentials). LDevID credentials are disabled by default and must be explicitly enabled before use; see 6.3.6 (Enable/disable DevID credential).

Inputs: The LDevID credential
Output: A value confirming that the DevID credential has been successfully stored. The credentialIndex is returned if the operation is successful. The keyIndex of the matching key.
This operation may be available via a remote management protocol.

6.3.13 LDevID Credential Chain insert

The DevID module should implement this operation. This operation allows the admistrator to insert a credential chain associated with a particular LDevID credential.

Inputs: The credentialIndex of the LDevID credential; and the chain associated with the credential.
Output: A value confirming that the credential chain has been stored.
This operation may be available via a remote management protocol.

[12]RFC 2986 refers to the *signature* of the certification request, which is equivalent to the RFC 5280 specified *signatureValue* for certificates.

6.3.14 LDevID Credential delete

The DevID module should implement this operation. This operation allows the administrator to delete a LDevID credential by specifying its credentialIndex. This operation implicitly performs an LDevID Credential Chain deletion. This operation does not remove the associated key material.

The DevID module may perform cryptographic zeroization on LDevID credential material as part of the credential delete process. Attempts to specify an IDevID credentialIndex shall result in a FALSE return value as the IDevID credential shall not be deleted.

Inputs: The credentialIndex
Output: A value confirming that the DevID credential has been removed.
This operation may be available via a remote management protocol.

6.3.15 LDevID Credential Chain delete

The DevID module should implement this operation. This operation allows the admistrator to delete a credential chain associated with a particular LDevID credential. This operation must occur before the credential itself is deleted.

Inputs: The credentialIndex of the LDevID credential this chain is currently associated with.
Output: A value confirming that the credential chain has been deleted. If there is no current credential chain, this operation returns success.
This operation may be available via a remote management protocol.

6.3.16 Addition of RNG entropy

The DevID module may implement this operation. This operation adds additional entropy to the RNG state to provide adequate entropy to support the desired security strength.

Inputs: Number of bytes of input, an array of <=256 bytes
Output: None
This operation may be available via a remote management protocol.

6.4 DevID Management Interface

This subclause defines management operations that provide support for centralized supervision of DevID devices. These facilities include inventory control, configuration and auditing capabilities.

Definition of a protocol to access the management operations is not in the scope. An SMIv2 MIB for these management operations is defined in Clause 8 (Management Information Base). A DevID module supporting SNMPv3 shall implement a management interface with authentication and privacy modes activated that supports the following operations.

Inventory control operations (see Table 6-3):

Table 6-3—Management Interface Inventory control operations

Obtain enumerated list of raw public keys	6.3.2
Obtain enumerated list of DevID credential information where for each credential the following are indicated: — sha1 hash of the certificate — certificate serial number — issuer — subject — HardwareModuleName — sha1 hash of the public key	6.3.3, and select fields from 7.2

Configuration operations (see Table 6-4):

Table 6-4—Management Interface Configuration operations

Enable/disable DevID credential	6.3.6
Enable/disable DevID key	6.3.7

Auditing operations (see Table 6-5):

Table 6-5—Management Interface Auditing operations

For each private key a count of how many signing operations	6.3.5
Count of key generations	6.3.8
Count of key insertions	6.3.9
Count of key deletions	6.3.10
Count of CSR operations	6.3.11
Count of credential insertions	6.3.12
Count of credential deletions	6.3.14

These operations have not been specified as part of the DevID Management Interface although they are appropriate for other management interfaces (see Table 6-6):

Table 6-6—DevID Service Interfaces that are not specified as part of the MIB

Enumeration of DevID credentials	6.3.3
Enumeration of DevID credential chains	6.3.4
LDevID key generation	6.3.8
LDevID key insertion	6.3.9
LDevID key delete	6.3.10
LDevID CSR generation	6.3.11
LDevID credential insertion	6.3.12
LDevID credential chain insertion	6.3.13
LDevID credential delete	6.3.14
LDevID credential chain delete	6.3.15
Addition of RNG entropy	6.3.16

6.5 PKI hierarchy requirements

Every DevID shall be unique within the scope of DevIDs issued by the CA responsible for a particular domain of use. DevIDs for distinct devices shall have an unique subject name as described in 7.2.8 (subject) and 7.2.9 (subjectAltName). Two DevIDs may be considered the same if they have the same issuer and subject name but different serial numbers (7.2.2).

The DevID module requirements imply infrastructure elements to obtain signed DevID credentials. The DevID Module communicates with a manufacturer's infrastructure for signing the IDevID. Communication with the manufacturer's CA can involve network communication between the manufacturing facility and any secure facilities chosen by the manufacturer to host their signing infrastructure. This can involve a wide area network (WAN), Virtual Private Network facilities, or complex Registration Authorities or other infrastructure elements.

Defining these mechanisms is out of scope of this standard. A lower security hurdle, and lower cost infrastructure, could be achieved by stationing a manufacturing CA in the manufacturing area. The CA Certification Practices Statement (CPS) as specified in RFC 3647 should be published in the PICS A.8 (DevID Supplied Information).

Similarly an LDevID credential implies the existence of a local CA. This can include a complex PKI hierarchy with Registration Authorities, etc., or a lower cost infrastructure that could be instantiated by single local CA. The appropriate cost and security trade-offs are out of scope of this standard.

The service interface discussed in 6.3 (DevID Service Interface) provides simple mechanisms sufficient for integration with whatever certificate generation infrastructure is chosen by the manufacturer/owners to create IDevID/LDevID certificates.

Figure 6-2 shows the DevID trust hierarchy. DevID credentials are issued by either the manufacturing CA infrastructure or a local CA. Each DevID credential provides a signed assertion of either the IDevID secret or an LDevID secret.

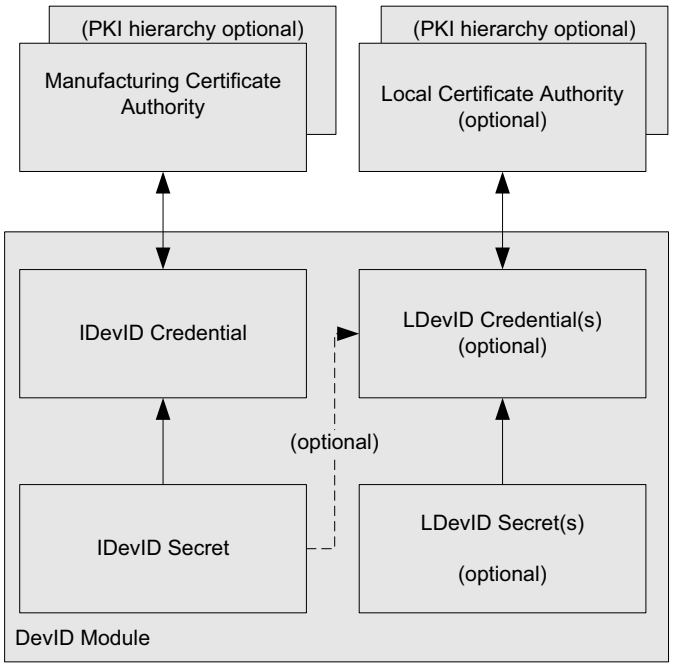

Figure 6-2—DevID trust hierarchy

6.5.1 Creation of DevID secret

The DevID secret can be generated internally in the DevID module, or can be generated by an external entity and securely installed in the DevID module. The manufacturer of the DevID module shall disclose whether IDevID key generation is internal or external to the DevID module.

NOTE—Externally generated keys that are not managed appropriately can compromise the security of the system. In particular, taking care to clear any residual key material for externally generated keys after insertion into the DevID module is strongly encouraged.

The IDevID secret may be shared between the IDevID and an LDevID, but the administrator of the device should be able to generate an independent LDevID secret to meet the identity hiding requirements indicated in 6.6.3 (Identity Hiding).

The quality of the key generated greatly influences the quality of the DevID credential. Poor randomness during key generation can lead to easily determined DevID secrets. The DevID secret should be generated within a DevID module using the RNG specified in 6.2.4 (Random number generator). If sufficient entropy is not available during manufacturing the IDevID secret may need to be generated externally.

The key generation mechanism and IDevID signing mechanisms associated with a particular DevID module should be reported in any CPS published for the issuer CA. The security inherent in the system used may be reflected in any subsequent certifications pursued for the devices in question, for example FIPS 140-2 [B31].

6.5.2 Creation of DevID credentials

To obtain an IDevID credential the DevID module shall first have a DevID secret as per the discussion in 6.5.1 (Creation of DevID secret). Generation and insertion of the IDevID credential itself is out of scope of this standard and is expected to occur during manufacturing.

To obtain an LDevID credential the DevID module shall first have a DevID secret as per the discussion in 6.5.1 (Creation of DevID secret). At that point a Certificate Signing Request (CSR) may be generated as specified in 6.3.11 (LDevID Certificate Signing Request generate). The CSR is transported to the local PKI hierarchy to be signed. The specifics of this enrollment operation are out of scope of this standard. The resulting LDevID credential is inserted into the DevID module using the "Insert LDevID Credential" operation.

6.5.3 Validation of DevIDs

The DevID is an X.509 credential and can be validated using the RFC 5280 defined mechanisms. IDevIDs are intended to have very long validity periods even exceeding what would normally be cryptographically acceptable. The manufacturer is not required to provide a Certificate Revocation List (CRL) although the validator may do CRL checking if the manufacturer provides CRLs. The validator may verify CRLs for LDevIDs as necessary.

In order to validate a DevID credential, the validation process must support the validation of the cryptographic primitives specified in 7.3 (Cryptographic Primitives) and the decoding of X.509 certificates using the distinguished encoding rules, as described in RFC 5280.

The protocols used for the exchange of information between the DevID module and the verifier are not specified by this standard. TLS is an example protocol that implements this exchange as discussed in C.1 (DevID use in EAP-TLS).

6.6 Trust Model

DevID credentials provide a method for network entities to authenticate each other. Authentication using DevIDs may be one-way (e.g., the network may control access of devices to services from the network), or mutual, in which case entities in the network authenticate themselves to each other. Authorization using DevIDs may limit the network(s) that a given device may obtain services from to only those network(s) that are specified in a solution-specific access control mechanism. For example a DevID enabled solution may include an access control list of device types, device serial numbers, LDevIDs, or other distinguishing credential values that are cryptographically bound to the credential and may be checked when making authorization decisions. The manufacturer shall describe in the PICS which distinguished subject name fields contain identifiers that are appropriate for authorization, auditing, or other purposes. The authentication and authorization mechanisms themselves are out of the scope of this standard.

The IDevID is signed by a credential that is held by the manufacturer and is part of a larger hierarchy. There is no global root and there are no requirements on the depth of the PKI hierarchy.

When asserting device identity, the LDevID should be used if it exists and is enabled. Enabling or disabling the DevID module functionalities as a whole shall always be under control of the device administrator.

Validation of a DevID credential provides authentication of the device in question. This is expected to be paired with an authorization mechanism.

When a DevID enabled network validates a DevID credential a DevID root credential store is used. Solutions that use the DevID to determine device identity are expected to support local configuration of the

DevID Root Credential Store. This enables the solution administrator to manipulate this root store by adding or removing root credentials for various manufacturers or LDevID devices that are authorized to be on their network. The solution administrator is expected to examine the CPS published for the device manufacturer's CA and the PICS proforma published for the DevID-compliant devices before adding manufacturer root credentials to their root store or authorization mechanism.

6.6.1 Mitigating risks of key exposure

If the manufacturer's signing credentials are compromised it would be possible for an attacker to create "fake" IDevID credentials. To mitigate this DevID enabled solutions are expected to either support installing LDevID credentials during deployment or to provide a mechanism to add IDevID public key specifics to the solution's authorization mechanisms. This limits any risk due to compromised root keys to only during DevID secured device deployment, which is a substantially smaller window of opportunity. The manufacturer is expected to use the same mechanisms they would use for any other high priority security vulnerability found in their product line when announcing potentially compromised signing credentials.

A compromised IDevID private key would allow an attacker to generate "clones" of the device. If any devices have the same keypair, even if they have different identities in their IDevIDs, they can spoof each other because they have knowledge of the same secret key. Use of a one-time authorization mechanism tied to LDevID enrollment prevents cloned or attacker devices from joining the network.

6.6.2 Security Objectives

DevIDs are used to authenticate access to networks. They do so using cryptographic data and operations. Some of this data is private. The DevID secret shall be stored and used within a controlled boundary (the *cryptographic boundary*) and protected from plaintext access by entities outside the cryptographic boundary. A DevID module shall satisfy the following security objectives:

a) *Confidentiality:* DevIDs contain one or more private cryptographic key(s), unique to that instance of DevID, that are to be protected from disclosure outside the cryptographic boundary. Failure to protect this data from external disclosure results in a compromised DevID that is no longer guaranteed to be bound to the device.
b) *Integrity:* The integrity of the public and private cryptographic keys and private identifier data is essential to ongoing use of DevIDs. Failure to protect the integrity of this data results in identification data that will no longer be useful for device authentication purposes.
c) *Availability:* A device compliant with the standard is shipped with an IDevID, which is available for device authentication via the runtime operations.
d) *Privacy:* The owner of the device can either disable the DevID module or use (multiple) LDevIDs to maintain privacy.

NOTE—Availability is prioritized over Privacy. The IDevID is enabled when the device is shipped and the owner has the ability to enable privacy by disabling the DevID module.

6.6.3 Identity Hiding

The IDevID shipped on a device could be used to track that exact device, which could violate the privacy of the device owner. For this reason, the device administrator shall be able to disable DevID module functionality through operations to disable access to all of the credentials or keys. Additionally the device administrator may wish to maintain the anonymity of the device but not disable the DevID functionality; to support this configuration the administrator shall be able to disable access to the IDevID private key and credential in favor of any indicated LDevID.

There is no DevID credential field that allows differentiation between an LDevID and an IDevID generated by an unknown manufacturer.

7. DevID Credential details

This clause describes the fields of the DevID credentials in detail. DevID credentials are a specified subset of RFC 5280 certificates. Solutions leveraging the DevID credential are expected to validate these credentials as specified in RFC 5280.

7.1 DevID hierarchy credential fields

The credentials used to build the PKI hierarchy (e.g., CAs) for DevID credentials shall conform to the general specification for DevID credentials indicated in 7.2 (DevID credential fields) with the following additions:

— The CA credentials shall include the “basic constraints” extension indicating the CAs of the PKI hierarchy are CAs as specified in RFC 5280. Use of critical extensions is discussed in more detail in 7.2.13 (extensions).
— The CA credentials shall include the “subjectKeyIdentifier” as specified in RFC 5280.

7.2 DevID credential fields

Table 7-1 summarizes the information presented in a DevID credential.

7.2.1 version

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

7.2.2 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.

7.2.3 signature

The identity of the signature algorithm used to sign the DevID certificate. The field is identical to the value of the signatureAlgorithm field [7.2.11 (signatureAlgorithm)].

7.2.4 issuer

The name of the signer of the DevID. This can be the name of the manufacturer, a third party CA, a network operator, a standards certification agency, or an end user organization.

The organization issuing the IDevID shall endeavor to 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 as specified in 7.2.5 (authorityKeyIdentifier), and the CA certificates in the DevID certificate chain contain the subjectKeyIdentifier extension as detailed in 7.1 (DevID hierarchy credential fields). Adding authorityKeyIdentifier and subjectKeyIdentifer facilitates certificate path building, which is necessary to validate DevID credentials. The Issuer field is predominately useful as a user interface mechanism.

Table 7-1—DevID credential fields summary

Field name	RFC 5280 type	Value	Subclause reference
version	INTEGER	v3	7.2.1
serialNumber	INTEGER	Positive integer	7.2.2
signature	AlgorithmIdentifier	RSASSA-PKCS1-v1_5 or ECDSA	7.2.3
issuer	Name	Name of issuing CA	7.2.4
authoritykeyidentifier	KeyIdentifier	A unique value that matches the subjectKeyIdentifier of the issuer's credential	7.2.5
notBefore	Time	Creation time of DevID	7.2.7.1
notAfter	Time	Expiry time of DevID	7.2.7.2
subject	Name	Name of the DevID device	7.2.8
subjectAltName	GeneralNames	Additional name elements for the DevID device	7.2.9
subjectPublicKeyInfo	SubjectPublicKeyInfo	The DevID public key	7.2.10
extensions	Extensions	Key usage and non-critical extensions	7.2.13
signatureAlgorithm	AlgorithmIdentifier	RSASSA-PKCS1-v1_5 or ECDSA signature	7.2.11
signatureValue	BIT STRING	DevID certificate signature	7.2.12

The LDevID issuer name is chosen by the administrator of the local PKI hierarchy and is expected to be unique within the local solution although 7.2.5 (authorityKeyIdentifier) and 7.1 (DevID hierarchy credential fields) still apply.

7.2.5 authorityKeyIdentifier

To optimize building the correct credential chain, the non-critical Authority Key Identifier extension shall be populated with a unique value as recommended by RFC 5280 and shall be included in all DevID certificates.

7.2.6 subjectKeyIdentifier

The Subject Key Identifier extension should not be included in DevID certificates.

NOTE—As specified in 7.1 (DevID hierarchy credential fields), the non-critical Subject Key Identifier extension of the CA certificates is populated with a unique value as recommended by RFC 5280. This facilitates certificate path building, which is necessary to validate DevID credentials. The DevID certificate itself does not contain the Subject Key Identifier extension both to conserve space and because it is not used when building certificate paths.

7.2.7 validity

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

All times shall be stated in the Universal Coordinated Time (UTC) time zone. Times up to and including 23:59:59 December 31, 2049 UTC shall be encoded as UTCTime as YYMMDDHHmmssZ. Times later than 23:59:59 December 31, 2049 UTC shall be encoded as GeneralizedTime as YYYYMMDDHHmmssZ.

7.2.7.1 notBefore

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

7.2.7.2 notAfter

The latest time a DevID is expected to be used.

Devices possessing an IDevID are expected to operate indefinitely into the future and should use the value 99991231235959Z.[13] Solutions verifying a DevID are expected to accept this value indefinitely. Any other value in a DevID notAfter field are expected be treated as specified in RFC 5280.

7.2.8 subject

The DevID subject field shall uniquely identify the device associated with the particular DevID credential within the issuer's domain of significance. The formatting of this field shall contain a unique X.500 Distinguished Name (DN). This may include the unique device serial number assigned by the manufacturer or any other suitable unique DN value that the issuer prefers. In the case of a third-party CA or a standards certification agency, this can contain the manufacturer's identity information.

The subject field's DN encoding should include the "serialNumber" attribute with the device's unique serial number.

7.2.9 subjectAltName

The non-critical DevID subjectAltName extension may supplement the subject field identity information as specified in RFC 5280 by containing a hardwareModuleName as specified in RFC 4108 [B22].

7.2.10 subjectPublicKeyInfo

The DevID subjectPublicKeyInfo field shall indicate the public key algorithm identifier and the public key in the specified algorithm format as specified in RFC 3279 [B10] and RFC 5480. The structure of the subjectPublicKeyInfo field is shown in Table 7-2.

The algorithm identifier for RSA keys is specified in RFC 3279 as rsaEncryption:

```
rsaEncryption OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }
```

RSA does not require the use of additional parameters; therefore, the parameters field is omitted. The RSAPublicKey format is as specified in RFC 3279 and included here for convenience:

```
RSAPublicKey ::= SEQUENCE {
  modulus            INTEGER,    -- n
  publicExponent     INTEGER  }  -- e
```

[13]This value corresponds to one second before the year 10 000; note the creation of an opportunity for the Y10K bug fix industry.

Table 7-2—subjectPublicKeyInfo data structure

For RSA:	subjectPublicKeyInfo ::= SEQUENCE {
	subjectPublicKeyAlgorithm Algorithm Identifier,
	publicKey RSAPublicKey }
	
	AlgorithmIdentifier ::= SEQUENCE {
	algorithmIdentifier "rsaEncryption",
	parameters "absent" }

For ECDSA:	subjectPublicKeyInfo ::= SEQUENCE {
	subjectPublicKeyAlgorithm AlgorithmIdentifier,
	publicKey ECPoint }
	
	AlgorithmIdentifier ::= SEQUENCE {
	algorithmIdentifier "id-ecPublicKey",
	parameters ECParameters}

The algorithm identifier for ECDSA signature keys is specified in RFC 5480 as id-ecPublicKey:

```
id-ecPublicKey  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2) us(840) ansi-x9-
62(10045) keyType(2) 1 }
```

ECDSA requires the use of certain parameters with the public key. As specified in RFC 5480 the parameter structure is:[14]

```
ECParameters ::= CHOICE {
       namedCurve         OBJECT IDENTIFIER
       -- implicitCurve   NULL
       -- specifiedCurve  SpecifiedECDomain
     }
```

The namedCurve choice, which allows all the required values for a particular set of elliptic curve domain parameters to be represented by an object identifier, shall be used. The object identifier for the P-256 curve is:

```
ansip256r1  OBJECT  IDENTIFIER  ::=  {  iso(1)  member-body(2)  us(840)  ansi-x9-
62(10045) curves(3) prime(1) 7 }
```

The ECDSA public key is as specified in RFC 5480 and included here for convenience:

```
ECPoint ::= OCTET STRING
```

Implementations shall support the uncompressed form and may support the compressed form of the ECDSA public key. The hybrid form of the ECC public key from ANSI X9.62-2005 shall not be used.

7.2.11 signatureAlgorithm

The DevID signatureAlgorithm field shall indicate the CA signature algorithm used to sign this certificate as specified in 7.3.1 for RSA and 7.3.2 for ECDSA.

[14]RFC 5480 updates the textual names of the parameters previously defined in RFC 3279 (e.g., EcpkParameters is ECParameters, ecParameters is specifiedCurve, implicitlyCA is implicitCA), although the underlying structures are consistent.

7.2.12 signatureValue

The CA's signature of the DevID certificate is computed using the CA's private 2048-bit RSA or 256-bit ECC certificate signing key using the algorithm identified in 7.2.11 (signatureAlgorithm).

When using RSA keys the DevID certificates shall be signed by the CA using the RSASSA-PKCS1-v1_5 algorithm identified in 7.3.1 (RSASSA-PKCS1-v1.5).

When using Elliptic Curve keys the DevID certificates shall be signed by the CA using the ECDSA algorithm identified in 7.3.2 (ECDSA). The structure for ECDSA signatures is as per RFC 5480 and is included here for convenience:

```
Ecdsa-Sig-Value ::= SEQUENCE {
  r INTEGER,
  s INTEGER }
```

7.2.13 extensions

The optional extensions field defined in RFC 5280 shall not be used for critical extensions in any IDevID credential with the exception of the keyUsage extension. The non-critical extensions included are specified in 7.2.5 (authorityKeyIdentifier) and 7.2.9 (subjectAltName).

The key usage field defined in RFC 5280 is intended to restrict the associated key material to only the specified usages. The intention of an IDevID is to provide a long lived credential useful for identifying the device in any future protocol uses. Restrictions applied during issuance may limit the future usefulness of the DevID. If a critical keyUsage extension is included in the IDevID, it shall include digitalSignature as defined in RFC 5280. The keyUsage extension may include keyEncipherment.

The optional extensions field defined in RFC 5280 should not be used for critical extensions in any LDevID credential. As specified in RFC 5280 a network solution that uses the LDevID credentials and encounters a critical extension it does not understand will signal an error and fail validation of the LDevID credential.

7.3 Cryptographic Primitives

7.3.1 RSASSA-PKCS1-v1.5

The RSASSA-PKCS1-v1.5 signature method is defined in RFC 3447. The algorithm shall be either sha1WithRSAEncryption or sha256WithRSAEncryption. The algorithm identifiers are:

```
sha1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2)us(840)
rsadsi(113549) pkcs(1) pkcs-1(1) 5 }

sha256WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2)us(840)
rsadsi(113549) pkcs(1) pkcs-1(1) 11 }
```

Support for sha1WithRSAEncryption is included for maximum interoperability but is not recommended. When the sha1WithRSAEncryption or sha256WithRSAEncryption algorithm identifiers appear in the algorithm field as an AlgorithmIdentifier, the encoding must omit the parameters field. That is, the AlgorithmIdentifier shall be a SEQUENCE of one component, the object identifier sha1WithRSAEncryption or sha256WithRSAEncryption.

7.3.2 ECDSA

The ECDSA signature method is defined in NIST FIPS 186-3 by reference to ANSI X9.62-2005. If implementing ECDSA, the SHA-256 message digest algorithm and the P-256 elliptic curve as defined in FIPS 186-3 Annex D, D.1.2.3, shall be used.

The signature algorithm shall be ecdsa-with-SHA256 as specified in RFC 5008. The algorithm identifier is:

```
ecdsa-with-SHA256  OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) ansi-X9-
62(10045) signatures(4) ecdsa-with-sha2(3) 2 }
```

When the ecdsa-with-SHA256 algorithm identifier appears in the algorithm field as an AlgorithmIdentifier, the encoding shall omit the parameters field. That is, the AlgorithmIdentifier shall be a SEQUENCE of one component, the object identifier ecdsa-with-SHA256.

7.3.3 SHA-256 hash algorithm

The hash algorithm used by the DevID module shall be the SHA-256 secure hash algorithm as defined in NIST FIPS 180-3.[15]

[15]The underlying ASN.1 structures fully support algorithm agility.

8. Management Information Base

This clause defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects for creating and managing secure device identifiers, based on the specifications contained in 6.3. This clause includes a MIB module that is compliant to SMIv2.

While this standard specifies MIB modules for use by DevID modules and entities that interact with them, there is no attempt here to define how the DevID module can be used to add security to SNMP.

8.1 Internet-Standard Management Framework

For a detailed overview of the standards that describe the current Internet-Standard Management Framework, refer to Section 7 of RFC 3410.

Managed objects are accessed via a virtual information store, termed the *Management Information Base* or MIB. MIB objects are generally accessed through the SNMP. Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in RFC 2578, RFC 2579, and RFC 2580.

8.2 Relationship to other MIB modules

8.2.1 Relationship to the Entity MIB

A system implementing this MIB module shall implement the entPhysicalTable defined in RFC 4133, the Entity MIB. The Entity MIB describes the managed objects used for managing multiple logical and physical entities managed by an SNMP agent. The entPhysicalIndex points to each physical device with DevID capability managed by an SNMP agent.

8.3 Structure of the MIB

A single MIB module is defined. Objects in the MIB are arranged into groups. Each group is organized as a set of related objects.

These objects are necessary to implement the DevID Management Interface described in Clause 6.4. The three MIB tables correspond to the groups of control operations described in 6.4 Table 6-3, Table 6-4, and Table 6-5.

Table 8-1 contains cross-references between the objects defined in Clause 6 and Clause 7 and the MIB objects defined in this clause.

Table 8-1—Managed object cross-reference table

MIB object(s)	Reference
DevID Mgmt Objects	
devIDPublicKeyCount	6.3.2
devIDPublicKeyTable	**6.3.2**
devIDPublicKeyIndex	6.3.2
devIDPublicKeyEnabled	6.3.2
devIDPublicKeyAlgorithm	7.2.10
devIDPublicKeyPubkeySHA1Hash	7.2.10
devIDPublicKeyErrStatus	Indicates an error of any key operations. See 6.4.
devIDCredentialTable	**6.3.3**
devIDCredentialIndex	6.3.3
devIDCredentialEnabled	6.3.3
devIDCredentialSHA1Hash	7.2
devIDCredentialSerialNumber	7.2.2
devIDCredentialIssuer	7.2.4
devIDCredentialSubject	7.2.8
devIDCredentialSubjectAltName	7.2.9
devIDCredentialPubkeyIndex	6.3.3
devIDCredentialErrStatus	Indicates an error of any credential operations. See 6.4.
devIDStatisticsTable	**6.4**
devIDStatisticKeyGenerationCount	6.4
devIDStatisticKeyInsertionCount	6.4
devIDStatisticKeyDeletionCount	6.4
devIDStatisticCSRGenerationCount	6.4
devIDStatisticCredentialInsertionCount	6.4
devIDStatisticCredentialDeletionCount	6.4

The DevID Service Interfaces described in 6.4 Table 6-6 are not included in the MIB.

8.4 Security considerations

SNMP versions prior to SNMPv3 did not include adequate security. SNMPv3 shall be used. Use of SNMPv3 without invocation of its cryptographic mechanisms (for authentication and privacy) is not consistent with the security goals of this standard. If those mechanisms are compromised, the management interface operations defined in 6.4 might be accessed.

There are management objects defined in this MIB module with a MAX-ACCESS clause of read-write. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. The objects and their sensitivity/vulnerability is described in Table 8-2.

Table 8-2—Sensitivity/vulnerability of read-write objects

MIB object(s)	Vulnerability
devIDPublicKeyEnabled	An enabled public key may expose the device's public key material. A disabled public key may restrict the device from identifying itself to a necessary resource.
devIDCredentialEnabled	An enabled credential may expose the device's credential information. A disabled credential may restrict the device from identifying itself to a necessary resource.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. The objects and their sensitivity/vulnerability is described in Table 8-3.

Table 8-3—Sensitivity/vulnerability of read-only objects

MIB object(s)	Vulnerability
devIDPublicKeyCount	Exposes network identity infrastructure details
devIDPublicKeyAlgorithm	Exposes the key algorithm
devIDPublicKeyPubkeySHA1Hash	Exposes identifying information
devIDPublicKeyErrStatus	Exposes errors providing potential indication of problems that could be used by an attacker
devIDCredentialCount	Exposes of network identity infrastructure details
devIDCredentialSHA1Hash	Exposes identifying information
devIDCredentialSerialNumber	Exposes identifying information
devIDCredentialIssuer	Exposes identifying information
devIDCredentialSubject	Exposes identifying information
devIDCredentialSubjectAltName	Exposes identifying information
devIDCredentialPubkeyIndex	Exposes identifying information

Table 8-3—Sensitivity/vulnerability of read-only objects (*Continued*)

MIB object(s)	Vulnerability
devIDCredentialErrStatus	Exposes errors providing potential indication of problems that could be used by an attacker
devIDStatisticKeyGenerationCount	Exposes network identity infrastructure details
devIDStatisticKeyInsertionCount	Exposes network identity infrastructure details
devIDStatisticKeyDeletionCount	Exposes network identity infrastructure details
devIDStatisticCSRGenerationCount	Exposes network identity infrastructure details
devIDStatisticCredentialInsertionCount	Exposes network identity infrastructure details
devIDStatisticCredentialDeletionCount	Exposes network identity infrastructure details

8.5 Definitions for Secure Device Identifier MIB

In the following MIB definition, if any discrepancy between the DESCRIPTION text and the corresponding definitions in Clause 6 or Clause 7 occur, the definitions shall take precedence.

```
--
************************************************************************
-- IEEE8021-DEVID-MIB
--
-- Definitions of managed objects supporting IEEE 802.1AR Secure Device
ID.
--
--
--
************************************************************************

IEEE8021-DEVID-MIB
DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32,
    Counter32
        FROM SNMPv2-SMI
    TruthValue,
    TEXTUAL-CONVENTION
        FROM SNMPv2-TC
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    PhysicalIndex, entPhysicalIndex
        FROM ENTITY-MIB;

ieee8021DevIDMIB MODULE-IDENTITY
```



```
    LAST-UPDATED    "200906250000Z"
    ORGANIZATION    "IEEE 802.1 Working Group"
    CONTACT-INFO
            "http:/grouper.ieee.org/groups/8021/index.html"
    DESCRIPTION
        "This is the management module of the Secure Device Identifier
        (DevID) for managing IEEE 802.1AR. As specified in the current
        IEEE 802.1AR draft document.
        This is the management module of the Secure Device Identifier
        (DevID) for managing IEEE 802.1AR. A DevID is cryptographically
        bound to a device, and supports authentication of the device's
        identity. Locally significant identities can be securely
        associated with an initial manufacturer-provisioned DevID and
        used in provisioning and authentication protocols to allow a
        network administrator to establish the trustworthiness of a
        device and select appropriate policies for transmission and
        reception of data and control protocols to and from the device.
        A device indicates any entity in an IEEE 802 LAN that seeks to
        obtain services from the network.
        The use of a Secure Device Identifier (DevID) in a DevID module
        allows network entities to associate a DevID credential with
        devices that participate or wish to participate in authenticated
        access to one or more networks.
        A device with DevID capability incorporates a globally unique
        manufacturer provided Initial SecureDevice Identifier (IDevID),
        stored in a way that ensures it will remain unmodified in the
        absence of both unrestricted access to the device and
        extraordinary efforts by an attacker.
        The device may support the creation of Locally Signification
        Device Identifiers (LDevIDs) by network administrator. Each
        LDevID is bound to the device in a way that makes it impossible
        for it to be forged or transferred to a device with a different
        IDevID without knowledge of the private key used to effect the
        crytographic binding. LDevID can incorporate, and fully protect,
        additional information specified by the network administrator to
        support local authorization conventions. LDevIDs may also be
        used to entirely replace IDevIDs in such a way as to assure the
        privacy of the user of a LDevID and the equipment in which it is
        installed.
        Every device has exactly one (IDevID) and zero or more LDevIDs.
        The number of LDevIDs depends upon the capabilities of the DevID
        module and on the administrative policy of the network(s) in
        which the device is used"
    REVISION        "200906250000Z"
    DESCRIPTION
        "Published as part of IEEE standard 802.1AR"
     ::= { iso (1) iso-identified-organization (3) ieee (111) standards-
association-numbered-series-standards (2) lan-man-stds (802)
ieee802dot1(1) ieee802dot1mibs(1) 17 }

devIDMIBNotifications  OBJECT IDENTIFIER   ::= { ieee8021DevIDMIB 0 }

devIDMIBObjects  OBJECT IDENTIFIER         ::= { ieee8021DevIDMIB 1 }
```

```
devIDMIBConformance  OBJECT IDENTIFIER     ::= { ieee8021DevIDMIB 2 }

--
-- Textual Convention
--
DevIDErrorStatus ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention is used to define the error state
        of a devID operation.

        The value none(1) which means no error, indicating a successful
       operation. The value internalError(2) is used to display an system
error."
    SYNTAX          INTEGER  {
                        none(1),
                        internalError(2)
                    }

DevIDAlgorithmIdentifier::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
       "This textual convention is used to define the algorithm type for
the public key."
    SYNTAX          INTEGER  {
                        rsaEncryption(1),
                        idecPublicKey(2)
                    }

--
-- Management Objects for DevID
--
devIDGlobalMIBObjects  OBJECT IDENTIFIER  ::= { devIDMIBObjects 1 }

devIDMgmtMIBObjects    OBJECT IDENTIFIER  ::= { devIDMIBObjects 2 }

devIDStatsMIBObjects  OBJECT IDENTIFIER   ::= { devIDMIBObjects 3 }


--
-- DevID global objects
--



--
-- DevID Mgmt Objects
--
devIDPublicKeyCount OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of DevID
        public keys installed in the DevID module."
```



```
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    ::= { devIDMgmtMIBObjects 1 }

--
-- DevID public key Table
--
devIDPublicKeyTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DevIDPublicKeyEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table containing the public key, the keys
        keyIndex, a value indicating if the key is
        enabled. This allows the administrator to
        determine the DevID keys installed in the
        DevID module. The maximum number of entries
        in this table is limited by the value of
        devIDPublicKeyCount."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    ::= { devIDMgmtMIBObjects 2 }

devIDPublicKeyEntry OBJECT-TYPE
    SYNTAX          DevIDPublicKeyEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry containing DevID public key,
        the keys keyIndex, a value indicating
        if the key is enabled"
    INDEX           { entPhysicalIndex }
    ::= { devIDPublicKeyTable 1 }

DevIDPublicKeyEntry ::= SEQUENCE {
        devIDPublicKeyIndex                         Unsigned32,
        devIDPublicKeyEnabled             TruthValue,
        devIDPublicKeyAlgorithm           DevIDAlgorithmIdentifier,
        devIDPublicKeyPubkeySHA1Hash    SnmpAdminString,
        devIDPublicKeyErrStatus           DevIDErrorStatus
}

devIDPublicKeyIndex OBJECT-TYPE
    SYNTAX          Unsigned32 (0..4294967295 )
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "All keys are indexed internally with this
        object. The value of this object is within
        0..devIDPublicKeyCount.
        This is the keyIndex and operations on
        keys will use the keyIndex to address a
        specific key. The IDevID key shall only be
        at index 0.
        Any error in retrieving a key will be displayed
        in the devIDPublicKeyErrStatus object."
```

```
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    ::= { devIDPublicKeyEntry 1 }

devIDPublicKeyEnabled OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The enable/disable state of this public key. This setting
                            persists across restarts."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    ::= { devIDPublicKeyEntry 2 }

devIDPublicKeyAlgorithm OBJECT-TYPE
    SYNTAX          DevIDAlgorithmIdentifier
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The DevID PublicKey Algorithm field shall indicate
        the public key algorithm identifier. This object
        identifies the public key algorithm as either
                            rsaEncryption or idecPublicKey"
    REFERENCE
        "IEEE 802.1AR 6.4, 6.3.2 and 7.2.9"
    ::= { devIDPublicKeyEntry 3 }

devIDPublicKeyPubkeySHA1Hash OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The SHA1 Hash of this DevID public key."
    REFERENCE
        "IEEE 802.1AR 6.4, 6.3.2 and 7.2.9"
    ::= { devIDPublicKeyEntry 4 }

devIDPublicKeyErrStatus OBJECT-TYPE
    SYNTAX          DevIDErrorStatus
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The displays the status of an operation on
        the public key. The default value is none
        which means no error, indicating a successful
        operation."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    DEFVAL          { none }
    ::= { devIDPublicKeyEntry 5 }


devIDCredentialCount OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
```



```
    DESCRIPTION
        "This gives the total number of DevID
        credentials installed in the DevID module."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    ::= { devIDMgmtMIBObjects 3 }

--
-- DevID Management Table
--
devIDCredentialTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DevIDCredentialEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table of current DevID credentials, where
        for each certificate the following are indicated:
        sha1 hash of the certificate, section7 defined
        fields of cert serial number, issuer, subject,
        HardwareModuleName, and pubkey."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.3"
    ::= { devIDMgmtMIBObjects 4 }

devIDCredentialEntry OBJECT-TYPE
    SYNTAX          DevIDCredentialEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry containing DevID Credential information"
    INDEX           { devIDCredentialIndex }
    ::= { devIDCredentialTable 1 }

DevIDCredentialEntry ::= SEQUENCE {
        devIDCredentialIndex                Unsigned32,
        devIDCredentialEnabled              TruthValue,
        devIDCredentialSHA1Hash             SnmpAdminString,
        devIDCredentialSerialNumber         SnmpAdminString,
        devIDCredentialIssuer               SnmpAdminString,
        devIDCredentialSubject              SnmpAdminString,
        devIDCredentialSubjectAltName       SnmpAdminString,
        devIDCredentialEntityIndex          PhysicalIndex,
        devIDCredentialPubkeyIndex          Unsigned32,
        devIDCredentialErrStatus            DevIDErrorStatus
}

devIDCredentialIndex OBJECT-TYPE
    SYNTAX          Unsigned32 (0..4294967295 )
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "All credentials are indexed internally with
        this object. The value of this object is within
        [0..devIDCredentialCount].
        This is the credentialIndex and operations on
        credentials will use the credentialIndex to address
```

```
        a specific crdential. The IDevID credential shall
        only be at index 0. Additional operations on credentials
        use the credentialIndex to address a specific
        credential."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    ::= { devIDCredentialEntry 1 }

devIDCredentialEnabled OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The enable/disable state of this credential. This setting
                        persists across restarts."
    REFERENCE       "IEEE 802.1AR 6.3.6"
    ::= { devIDCredentialEntry 2 }

devIDCredentialSHA1Hash     OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The SHA1 Hash of this DevID credential."
    REFERENCE       "IEEE 802.1AR 7.2.2"
    ::= { devIDCredentialEntry 3 }

devIDCredentialSerialNumber OBJECT-TYPE
    SYNTAX          SnmpAdminString (SIZE  (0..20))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The serial number of the credential."
    REFERENCE       "IEEE 802.1AR 7.2.2"
    ::= { devIDCredentialEntry 4 }

devIDCredentialIssuer OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The issuer field of the credential."
    REFERENCE       "IEEE 802.1AR 7.2.4"
    ::= { devIDCredentialEntry 5 }

devIDCredentialSubject OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The subject field of the credential."
    REFERENCE       "IEEE 802.1AR 7.2.8"
    ::= { devIDCredentialEntry 6 }

devIDCredentialSubjectAltName OBJECT-TYPE
```



```
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The subjectaltname field of the credential"
    REFERENCE       "IEEE 802.1AR 7.2.8"
    ::= { devIDCredentialEntry 7 }

devIDCredentialEntityIndex    OBJECT-TYPE
    SYNTAX          PhysicalIndex
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
            "This refers to the entPhysicalIndex in
            entPhysicalTable to identify the associated
            physical entity."
     REFERENCE       "IEEE 802.1AR 6.4"
    ::= { devIDCredentialEntry 8 }

devIDCredentialPubkeyIndex OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Has the appropriate devIDPublicKeyIndex
        value from devIDPublicKeyTable to identify
        the public key information."
    REFERENCE       "IEEE 802.1AR 7.2.9"
    ::= { devIDCredentialEntry 9 }

devIDCredentialErrStatus OBJECT-TYPE
    SYNTAX          DevIDErrorStatus
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The displays the status of an operation on
        the credential. The default value is none
        which means no error, indicating a successful
        operation."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.2"
    DEFVAL          { none }
    ::= { devIDCredentialEntry 10 }


--
-- DevID statistics table
--
devIDStatisticsTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DevIDStatisticsEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table containing statistics information."
    ::= { devIDMgmtMIBObjects 5 }
```

```
devIDStatisticsEntry OBJECT-TYPE
    SYNTAX          DevIDStatisticsEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry containing DevID statistics."
    INDEX           { entPhysicalIndex }
    ::= { devIDStatisticsTable 1 }

DevIDStatisticsEntry ::= SEQUENCE {
        devIDStatisticKeyGenerationCount         Counter32,
        devIDStatisticKeyInsertionCount          Counter32,
        devIDStatisticKeyDeletionCount           Counter32,
        devIDStatisticCSRGenerationCount         Counter32,
        devIDStatisticCredentialInsertionCount   Counter32,
        devIDStatisticCredentialDeletionCount    Counter32
}

devIDStatisticKeyGenerationCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of additional
        LDevID key material generation within the
        DevID module.
                            Discontinuities occur at system restart and
counter rollover."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.8"
    ::= { devIDStatisticsEntry 1 }

devIDStatisticKeyInsertionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of additional
        LDevID key material generation followed by
        an insertion within the DevID module.
                            Discontinuities occur at system restart and
counter rollover."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.9"
    ::= { devIDStatisticsEntry 2 }

devIDStatisticKeyDeletionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of LDevID keypair
        deletions within the DevID module.
                            Discontinuities occur at system restart and
counter rollover."
```



```
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.10"
    ::= { devIDStatisticsEntry 3 }

devIDStatisticCSRGenerationCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of
        Certificate Signing Request (CSR)
        generation as defined in RFC2986.
                            Discontinuities occur at system restart and
counter rollover."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.11"
    ::= { devIDStatisticsEntry 4 }

devIDStatisticCredentialInsertionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of LDevID
        credentials installed into the DevID module.
                            Discontinuities occur at system restart and
counter rollover."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.12"
    ::= { devIDStatisticsEntry 5 }

devIDStatisticCredentialDeletionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This gives the total number of LDevID
        credentials deletions into the DevID module.
                            Discontinuities occur at system restart and
counter rollover."
    REFERENCE       "IEEE 802.1AR 6.4, and 6.3.14"
    ::= { devIDStatisticsEntry 6 }

--
-- Conformance Information Definition
--
devIDMIBCompliances  OBJECT IDENTIFIER
    ::= { devIDMIBConformance 1 }

devIDMIBGroups  OBJECT IDENTIFIER
    ::= { devIDMIBConformance 2 }

devIDMIBModuleCompliance MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "Module Compliance for this mib."
    MODULE          -- this module
```

```
    MANDATORY-GROUPS {
                        devIDMIBObjectGroup
                    }
    ::= { devIDMIBCompliances 1 }

devIDMIBObjectGroup OBJECT-GROUP
    OBJECTS         {
                        devIDPublicKeyCount,
                        devIDPublicKeyEnabled,
                        devIDPublicKeyAlgorithm,
                        devIDPublicKeyPubkeySHA1Hash,
                        devIDPublicKeyErrStatus,
                        devIDCredentialCount,
                        devIDCredentialEnabled,
                        devIDCredentialSHA1Hash,
                        devIDCredentialSerialNumber,
                        devIDCredentialIssuer,
                        devIDCredentialSubject,
                        devIDCredentialSubjectAltName,
                        devIDCredentialEntityIndex,
                        devIDCredentialPubkeyIndex,
                       devIDCredentialErrStatus,
                        devIDStatisticKeyGenerationCount,
                        devIDStatisticKeyInsertionCount,
                        devIDStatisticKeyDeletionCount,
                        devIDStatisticCSRGenerationCount,
                        devIDStatisticCredentialInsertionCount,
                        devIDStatisticCredentialDeletionCount
}
    STATUS          current
    DESCRIPTION
        "A collection of objects providing public key
        manageability, credential manageability and stats."
    ::= { devIDMIBGroups 1 }

END
```


Annex A

(normative)

PICS Proforma[16]

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to this standard shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of the capabilities and options of the protocol that have been implemented. The PICS can have a number of uses, including use

a) By the protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight.
b) By the supplier and acquirer—or potential acquirer—of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma.
c) By the user—or potential user—of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that although interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSs).
d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A.2 Abbreviations and special symbols

A.2.1 Status symbols

M Mandatory
O Optional
O.n Optional, but support of at least one of the group of options labeled by the same numeral *n* is required
X Prohibited
pred: Conditional-item symbol, including predicate identification (see A.3.4)
¬ Logical negation, applied to a conditional item's predicate

A.2.2 General abbreviations

N/A Not applicable
PICS Protocol Implementation Conformance Statement

[16]*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

A.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

The first part of the PICS proforma, implementation identification and protocol summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire, divided into several subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No), or by entering a value or a set or range of values. (Note that there are some items in which two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered; the third column records the status of the item—whether support is mandatory, optional, or conditional; see also A.3.4. The fourth column contains the reference or references to the material that specifies the item in the main body of this standard, and the fifth column provides the space for the answers.

A supplier may also provide (or be required to provide) further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled *Ai* or *Xi,* respectively, for cross-referencing purposes, where *i* is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformance Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, in case that makes for easier and clearer presentation of the information.

A.3.2 Additional Information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing on the answers to some items.

References to items of Additional Information may be entered next to any answer in the questionnaire and may be included in items of Exception Information.

A.3.3 Exception Information

It may occasionally happen that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this; instead, the supplier shall write the missing answer into the Support column, together with an *Xi* reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item.

An implementation for which an Exception item is required in this way does not conform to this standard.

NOTE—A possible reason for the preceding situation is that a defect in this standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional status

A.3.4.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself and its status if it does apply—mandatory or optional—are dependent upon whether or not certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the Not Applicable answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form "**pred:** S" where **pred** is a predicate as described in A.3.4.2, and S is a status symbol, M or 0.

If the value of the predicate is true (see A.3.4.2), the conditional item is applicable, and its status is indicated by the status symbol following the predicate: the answer column is to be marked in the usual way. If the value of the predicate is false, the Not Applicable (N/A) answer is to be marked.

A.3.4.2 Predicates

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma: The value of the predicate is true if the item is marked as supported, and is false otherwise.
b) A predicate-name, for a predicate defined as a boolean expression constructed by combining item-references using the boolean operator OR: The value of the predicate is true if one or more of the items is marked as supported.
c) A predicate-name, for a predicate defined as a boolean expression constructed by combining item-references using the boolean operator AND: The value of the predicate is true if all of the items are marked as supported.
d) The logical negation symbol "¬" prefixed to an item-reference or predicate-name: The value of the predicate is true if the value of the predicate formed by omitting the ¬ symbol is false, and vice versa.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

A.4 PICS proforma for IEEE 802.1AR

A.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names	
NOTE—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).	

A.4.2 Protocol summary, IEEE Std 802.1AR-2009

Identification of protocol specification	IEEE Std 802.1AR-2009, IEEE Standard for Local and Metropolitan Area Networks—Secure Device Identity
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	Amd. : Corr. : Amd. : Corr. :
Have any Exception items been required? (See A.3.3: The answer Yes means that the implementation does not conform to IEEE Std 802.AR-2009.)	No [ ] Yes [ ]

Date of Statement	

A.5 Major capabilities and options

Item	Feature	Status	References	Support
	Does the implementation:			
IDEV	Contain an Initial Device Identifier (IDevID)	M	6.2.1	Yes [ ]
DEVIDF	Include specified fields in all DevIDs	M	6.2.1	Yes [ ]
CHAIN	Contain an IDevID certificate chain	M	6.2.1	Yes [ ]
KEYRSA	Support RSA asymmetric cryptography	O.1	6.2.2	Yes [ ] No [ ]
KEYECDSA	Support ECDSA asymmetric cryptography	O.1	6.2.2	Yes [ ] No [ ]
STOR	Meet the minimum requirements for secure storage	M	6.2.5	Yes [ ]
DEVSI	Meet the minimum requirements for the service interface	M	6.3	Yes [ ]
EDEVSI	Provide for some or all of the LDevID service interface functionalities	O	6.3	Yes [ ] No [ ]
CHAINF	Include specified additional fields for DevID certificate chains	M	7.1	Yes [ ]
SERIALNUM	Does the subject name field include the unique device serial number?	O	7.2.8	Yes [ ] No [ ]
HWMODNAME	Does the subjectAltName field include the device hardwareModuleName	O	7.2.9	Yes [ ] No [ ]
SNMPV3	Support access to the MIB using SNMP v3	O	Clause 8	Yes [ ] No [ ]

A.6 DevID Service Interface

Item	Feature	Status	References	Support
SI-1	Initialization	M	6.3.1	Yes [ ]
SI-2	Enumeration of the DevID public keys	M	6.3.2	Yes [ ]
SI-3	Enumeration of the DevID credentials	M	6.3.3	Yes [ ]
SI-4	Enumeration of the DevID credential chain	M	6.3.4	Yes [ ]
SI-5	Signing	M	6.3.5	Yes [ ]
SI-5	Enable/Disable DevID credential	M	6.3.6	Yes [ ]
SI-7	Enable/Disable DevID key	M	6.3.7	Yes [ ]
SI-8	Enumeration of the DevID credential chain	M	6.3.4	Yes [ ]
SI-9	LDevID Key generation	O	6.3.8	Yes [ ] No [ ]
SI-10	LDevID Key insert	O	6.3.9	Yes [ ] No [ ]
SI-11	LDevID Key delete	O	6.3.10	Yes [ ] No [ ]
SI-12	Is it possible to remove the IDevID	X	6.3.10	No [ ]
SI-13	LDevID Certificate Signing Request generation	O	6.3.11	Yes [ ] No [ ]
SI-14	LDevID Credential insert	O	6.3.12	Yes [ ] No [ ]
SI-15	LDevID Credential chain insert	O	6.3.13	Yes [ ] No [ ]
SI-16	LDevID Credential delete	O	6.3.14	Yes [ ] No [ ]
SI-17	Addition of RNG entropy	O	6.3.16	Yes [ ] No [ ]

A.7 DevID Management Interface

Item	Feature	Status	References	Support
MI-1	Enumerated list of raw public keys	SNMPV3:M	6.4	Yes [ ]
MI-2	Enumeration of the DevID credentials	SNMPV3:M	6.4	Yes [ ]
MI-3	Enable/Disable DevID credential	SNMPV3:M	6.4	Yes [ ]
MI-4	Enable/Disable DevID key	SNMPV3:M	6.4	Yes [ ]
MI-5	For each key a count of Signing operations	SNMPV3:M	6.4	Yes [ ]
MI-6	For each key a count of LDevID Certificate Signing Request generations	SNMPV3:M	6.4	Yes [ ]
MI-7	A count of LDevID Key generations	SNMPV3:M	6.4	Yes [ ]
MI-8	A count of LDevID Key insertions	SNMPV3:M	6.4	Yes [ ]
MI-9	A count of LDevID Key deletions	SNMPV3:M	6.4	Yes [ ]

A.8 DevID Supplied Information

Item	Feature	Status	References	Support
ENTITY	What device is being identified		6.1	S_
ESTOR	What encryption mechanism is used if external storage is supported	O	6.2.5	S_
KEYEXP	What notification method will be used if a credential signing key is suspected of being compromised	M	6.6.1	S_
KEYGEN	What are the key generation mechanisms and associated IDevID signing mechanisms for the IDevID credential	M	6.5.1	S_
ISSUER	What is the basis for the belief that the issuer name is unique	M	7.2.4	S_
CPS	What is the CA's CPS	O	6.5	S_
AUTH	What distinguished subject name fields contain identifiers that are appropriate for authorization, auditing, or other purposes. How are these parsed?	M	6.6	S_

Annex B

(normative)

Implementing a DevID with a TPM

The Trusted Computing Group (TCG) has defined a Trusted Platform Module (TPM), which provides a device with a "root of trust" for security services. Central to the capabilities of a TPM is the notion of device identity and the support of multiple cryptographic keys that can be bound to the device and used in communication protocols to authenticate that device.

The TPM specification version 1.2 is available from the Trusted Computing Group.[17] The TPM specification provides support for the core capabilities of a DevID module as defined by this standard, and this appendix aims to review how those capabilities match DevID module goals and requirements. There may be multiple ways of implementing a DevID module using the capabilities of a TPM, and this annex will discuss some of the more important aspects in using a TPM as the basis for implementing a DevID module.

B.1 DevID goals

The goals of a DevID module are discussed in the purpose and scope of this standard. In summary these include specifying a unique per-device DevID that ships with the device as an initial credential, protecting the binding of this DevID to the device, defining a means of adding a locally significant credential, and using the credential in authentication exchanges. The TPM meets these goals in the following ways.

B.1.1 Initial DevID

DevID compliant devices have exactly one Initial Secure Device Identifier (IDevID). This includes a DevID Credential and a DevID Secret that are used inside of the module to prove identity of the device during authentication exchanges.

A TPM can be procured with an Endorsement Key (EK) and associated Endorsement Certificate signed by a trusted third party (typically the TPM manufacturer). This provides a unique credential that is bound to the device in which a TPM is embedded. This specifically allows an authenticator to establish trust remotely in a TPM post-deployment operation in the field, and to further establish trust in the properties of TPM keys generated post-deployment in order to certify them online and remotely.

However the TCG specifications do not allow for an EK to be used as an identity for authentication purposes (i.e., digital signature operations) because of privacy concerns. Therefore a device manufacturer that would want to include an initial identity in the device for digital signature purposes would have to initialize the TPM and create a further identity credential during the manufacturing process. In this situation, the TPM would be controlled by the manufacturer, and mechanisms within the device are needed to prohibit the user from reinitializing the TPM. This process is different from the typical model used by the TCG for PC Client TPMs where the owner of the equipment initializes the TPM.

B.1.2 Protecting the DevID and its binding to the device

A DevID module is cryptographically bound to a device using cryptographic operations to combine a secret with other specified data objects such that it may be proven that the result could only be created by an entity

[17] Available at http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

having knowledge of the secret. The DevID secret is securely protected from access by any other entity and kept within the cryptographic boundary of the DevID module.

A TPM can generate and/or protect cryptographic keys and control their use for digital signature operations. In particular a TPM can control the keys that it generates and can prevent them from ever being available outside of the TPM itself. This supports the implementation of a DevID module with a TPM in a way that uniquely binds a cryptographic identity to a device in which that TPM is embedded.

B.1.3 Adding locally significant credentials

A DevID compliant device may optionally support the creation of Locally Significant Device Identifiers (LDevIDs) by a network administrator. Each LDevID is bound to the device and cannot be forged or transferred to another device without the knowledge of the private key used to create the cryptographic binding.

The TPM supports the creation and usage of an unlimited number of protected non-migratable asymmetric key pairs. These key pairs can be used to support the creation of multiple TPM-protected locally significant credentials. Non-migratable asymmetric key pairs are locked to a given TPM, are never duplicated, and must be created by the TPM within the cryptographic boundary. These key pairs are the basis for the LDevID secret and are bound to the TPM through the AIK key creation process.

B.1.4 Using the DevID credential in authentication exchanges

DevIDs are designed to be used as authentication credentials with industry standard authentication protocols such as the Extensible Authentication Protocol (EAP). The validation of a DevID credential relies upon particular cryptographic operations supported by the module and in particular the capabilities of the mandatory signing operation.

TPMs are used today in a number of different industry standard authentication protocols including EAP. However, specific usage of a TPM key in an authentication protocol requires compatibility of the cryptographic algorithms used in the protocol. DevID module cryptographic algorithms are discussed in 6.2.2.

B.2 DevID requirements

The requirements of a DevID module are described in Clause 5. These include providing storage for the DevID secret and credential, RSA or ECC key pairs, SHA-256 hashing, a random number generator, and support for mandatory operations. The TPM meets these requirements in the following ways.

B.2.1 Storage

DevIDs are required to be stored on the device in a way to prevent the removal of the identifier from the device and allowing use with another device. In particular the DevID secret is stored in a protected fashion.

Protected storage of secrets is a core function of the TPM. It supports the creation, import, and controlled usage of an unlimited number of asymmetric cryptographic keys in a single TPM. The asymmetric keys can be stored internally within the TPM in a small, non-volatile storage area or they may be wrapped and stored externally. The TPM uses a storage root key that is always internal to the module to protect secrets using asymmetric cryptography. An unlimited number of additional asymmetric storage keys may be created.

B.2.2 Asymmetric secret

The DevID secret must be either an RSA private-public key pair of length 2048 bits or a P-256 Elliptic Curve private-public key pair.

The TPM requires the support of the RSA algorithm up to 2048-bit key sizes. The TPM does not currently support ECC algorithms.

B.2.3 Hashing

DevID specifies that the hash algorithm used for all credentials shall be SHA-256.

The TPM currently supports SHA-1 as a required hash algorithm. However, it is possible for the user of the TPM to generate and insert a different hash for signing operations. In order to support the DevID requirement of hashing with SHA-256, the device using the TPM must use the PKCSv1 DER encoding interface and insert an externally generated SHA-256 hash.

B.2.4 Random number generator

DevID specifies an internal RNG with 128-bit strength that meets the requirements of NIST Special Publication 800-90 [B33]. The RNG is used for signature generation, key generation, and cryptographic authentication protocols.

The TPM predates the NIST specifications, but can be enabled to meet the requirements. The TPM includes a command that allows the addition of entropy to its RNG engine, which can be used in such a way to seed the RNG as required. The TPM has a general purpose interface to retrieve random numbers from the RNG, which may be useful for devices implementing DevIDs and using them in authentication protocols.

B.2.5 Service interface

The DevID module specification provides a service interface that enables operations and control over the device's identity. The device itself should follow the principle of least privilege and limit access to these operations in such a way to support the device's administrative model. The following operations are specified.

B.2.5.1 Initialize

The DevID module needs to be reset, initialized, and enabled for run-time operation. It may be possible to add additional entropy to the RNG during this process.

The process of initializing a TPM involves several steps that must be protected and precede the equivalent steps of initializing a DevID. A factory fresh TPM must be initialized, enabled, and taken ownership of. Taking ownership involves setting the TPM ownership authorization data, used to control and authorize key TPM commands and functions. Since the IDevID is intended to be permanently bound to the device, the owner of a TPM implementing an IDevID must protect the interface to TPM clear operations so that the contents of the TPM are not erased. Once owned, the TPM power-on start-up procedures and TPM enable/disable functions support the requirements of a DevID initialization operation. It is possible to include input to the RNG entropy.

B.2.5.2 Enumeration

The DevID module supports enumeration of the DevID public keys, credentials, and credential chains. The firmware managing the TPM can support these operations using the appropriate TPM interfaces to retrieve relevant data.

B.2.5.3 Signing

The DevID module requires the support of a signing operation that uses the available cryptographic functions exclusively by the local device during authentication protocols. The cryptographic functions include public key cryptography using RSA or ECC keys and SHA-256 hashing. The signing operation should not be allowed to be executed remotely.

The TPM is typically used with the RSA RSASSA-PKCS1-v1.5 signature algorithm including SHA-1. However, the TPM further specifies how to format the input of the signature operation to allow alternative hash functions. It will be necessary to use the RSA PKCS1v1.5_DER scheme to achieve compatibility with the DevID requirement for hashing with SHA-256. ECC is currently not supported by the TPM.

The local controlling firmware is responsible for preventing remote execution of the signing operation.

B.2.5.4 Enable/disable keys and credentials

Subclause 6.3 specifies the ability to enable and disable individual DevID keys and/or credentials, thus preventing the device from asserting that particular identity during authentication exchanges.

Individual DevIDs implemented by a TPM module are managed by the operating firmware interfacing to the TPM. Since a DevID secret and credential involve an active key hierarchy within the TPM, it will be necessary for the operating firmware to present individual DevID enable/disable operations. The TPM itself supports enable and disable operations, but these will globally enable or disable the TPM and should not be exposed to the user of the device.

B.2.5.5 Key pair generation

The DevID module supports the creation, addition, and deletion of key pairs in order to provision and revoke LDevIDs. Key pairs are either created internally or passed in through a secure interface. Key pairs may also be removed from the DevID module to support zeroization.

The TPM supports the creation, addition, and deletion of RSA key pairs. Typically key pairs are internally generated, but they may also be passed into the TPM using a wrapping function that is secured by a TPM parent key. TPM version 1.2 supports the concept of transport protection that facilitates the establishment of a secure channel between the TPM and secure processes to provide confidentiality and integrity protection of commands sent to the TPM. Transport security can be used to authorize critical operations such as the deletion of LDevID key pairs.

B.2.5.6 Certificate Signing Request

DevID modules support the ability to generate a Certificate Signing Request. The firmware managing the TPM can support this operation.

B.2.5.7 Insertion and deletion of credentials and credential chains

LDevID credentials and their associated chains are installed or removed from the DevID module using the insert and delete operations. An existing key pair is associated with the credential during these operations, but the key pair is managed separately.

There is minimal non-volatile storage within the TPM itself, so credentials and certificates will typically be stored externally. The TPM can protect this external storage using the sealing functions if desired, or the controlling firmware of the DevID implementation can support these operations directly without cryptographic support.

Annex C

(informative)

Scenarios for DevID

DevID credentials are expected to be used during cryptographic identification and authentication protocols. As indicated in 1.1, this standard describes interfaces and methods for the use of DevIDs with existing provisioning and authentication protocols. It does not provide a normative definition of such a protocol.

This annex specifies how to use DevID for common device identification scenarios.

C.1 DevID use in EAP-TLS

Identification of network devices is often by an IEEE 802.1X EAP method. Many EAP methods, such as EAP_TLS, use X.509 client credentials and thus a DevID could be used. Additionally IEEE Std 802.1X [B2] indicates EAP-TLS support for devices that support this standard.

A device that uses a DevID module with RSA-based credentials for client identity during an EAP-TLS exchange can be expected to use the operations defined in 6.3 (DevID Service Interface) as follows:

a) 6.3.3 (Enumeration of DevID credentials) to determine the available and enabled DevID credentials. How the EAP-TLS implementation chooses the appropriate DevID credential is solution specific.
b) 6.3.5.1 (RSA Signing) to perform the signing operation during the TLS handshake.

Because the DevID module only supports a signing operation, the TLS cipher suite must be TLS_DHE_RSA_WITH_AES_128_CBC_SHA or one of the ECDSA cipher suites listed in RFC 5289.

A simplified EAP-TLS mutual authentication negotiation is shown in Figure C.1. For simplicity, assume that the authenticator also provides the authentication service. In the second exchange of the protocol, the authenticating peer identifies itself with a name. This name might, or might not, be related to the identity it will present during the TLS negotiation. In the fourth exchange of the protocol, the peer sends a client_hello to the authenticator. Part of this message is a random number (“nonce”) that the client chooses to detect attempts to deceive it by substituting, altering, or replaying messages during the negotiation. This highlights an important feature of the DevID module: the provision of a cryptographic quality random number generator for use in the protocol. In the fifth exchange, the server responds with its own nonce, a copy of its DevID certificate including its unique name and public cryptographic key, and a request for the client to send its unique DevID certificate. After receiving server_hello_done, the client responds with its DevID certificate and certificate_verify message to prove possession of the DevID secret key. The client then installs the newly derived symmetric key, turns on encryption on its transmission channel, and sends the authenticator its finished message (over the secure channel) that includes a secure hash of the messages in the exchange. The authenticator derives and installs the encryption keys and turns on decryption in its receiver. The contents of the decrypted finished message validate that all of the messages the client sent and received were authentic (received as-sent, without error). If validation is successful, the authenticator can trust the authenticity of all subsequent data sent by the client over the secure channel. It now sends a change_cipher_spec message to the client and enables encryption on its transmission channel. The authenticator computes the hash of all messages it sent and received throughout the handshake protocol with its authenticating peer and sends those in its finished message over the encrypted channel. Both parties have now identified and validated the identity of each other using the DevID certificate and DevID secret.

The use of EAP-TLS or a similar EAP method is expected but not mandated by this standard.

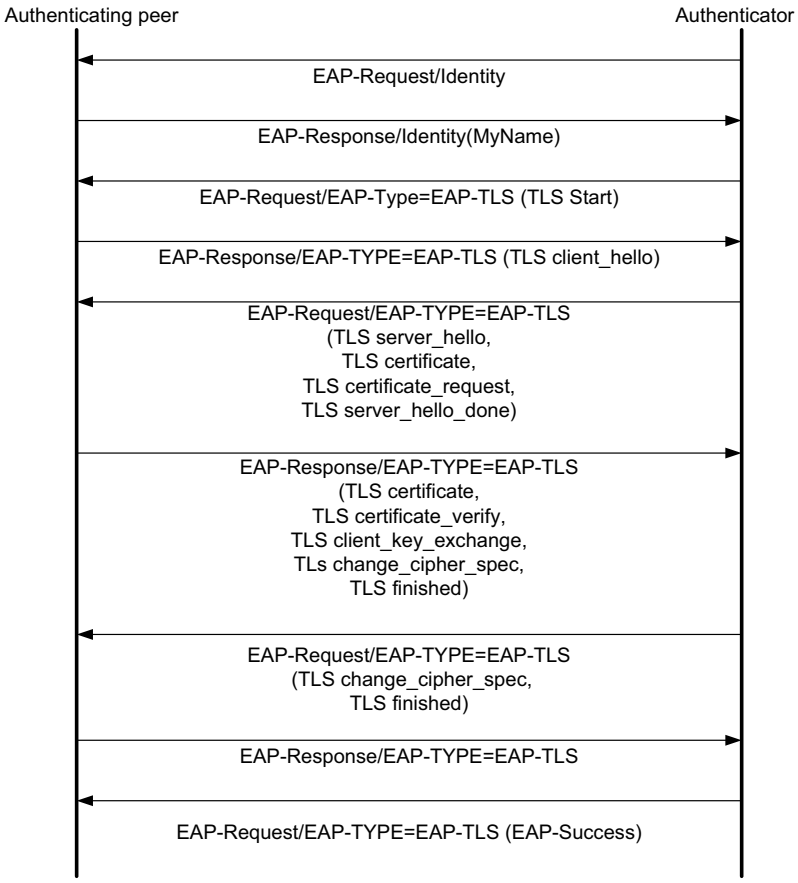

Figure C.1—Example EAP-TLS exchange

C.2 DevID uses in consumer devices

End users are not expected to directly use the IEEE 802.1AR device identity. Instead the DevID is expected to be used by a home networking access point or router. These devices provide network connectivity to all devices on the home network and also provide security mechanisms such as passwords or Web-based authentication.

Often routers and access points also include a mechanism for limiting which devices can join the network through configuration of a list of allowed MAC addresses or other device identifying information. The IEEE 802.1AR device identity can be used as a secure form of identity for these purposes. Vendors that include human readable identity information within the DevID subject field, or use a machine readable serialNumber attribute, or the subjectAltName hardwareModuleName, can provide integrated solutions with interfaces that are both more user-friendly and more secure than current MAC address-based solutions.

C.3 DevID uses in enterprise devices

In an enterprise environment the IEEE 802.1AR device identity is expected to be used to enroll the device into a local security infrastructure. During enrollment the device only needs to be authorized via a simple procedure, thus reducing the costs associated with having trained staff deploying all devices. Similar to the

consumer-use space, the existence of a human-readable and easily understood device-identifier bound to a cryptographic identity provides a building block for this type of usage.

An illustrative example is an IEEE 802.1X network with strict authentication and authorization for devices on the corporate network leveraging a central database such as an AAA server. Administration of this database may be restricted to system administrators or it may be exposed to general employees using a Web page or other common technology, such that this central database contains the approved manufacturers and a list of the DevIDs of authorized devices. Authorizing new devices to be on the network can be done by inserting human readable DevID subject-name information into this central database. This can occur while devices are off-line. For example, the DevID subject-name can be listed on the bill of sale (for devices being drop shipped to a final destination) so that database entries can be entered on completion of the order. Alternatively, it can be done by employees entering the information into the database via a configuration Web page after purchasing an approved device from a retail establishment.

The distinctions between this and the consumer use case predominately have to do with the authorization interface on the “router or access point” infrastructure. In the consumer use case this is envisioned as a relatively simple Web-driven interface, whereas in the enterprise environment a backend AAA server and potentially complicated “importing of bill of sale information” is supposed.

Annex D

(informative)

Bibliography

[B1] Fowler, M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language,* Third Edition. Boston: Pearson Education Inc., 2004.

[B2] IEEE Std 802.1X™, IEEE Standard for Local and metropolitan area networks—Port-Based Network Access Control.[18, 19]

[B3] IEEE Std 802.2™-1998 [ISO/IEC 8802-2:1998], Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 2: Logical link control.

[B4] IEEE Std 1363™-2000, IEEE Standard Specifications for Public-Key Cryptography.

[B5] IETF RFC 787, Connectionless data transmission survey/tutorial, A. Lyman Chapin, July 1981.[20]

[B6] IETF RFC 2104, HMAC: Keyed-Hashing for Message Authentication, H. Krawczyk, M. Bellare, and R.Canetti, February 1997.

[B7] IETF RFC 2279, UTF-8, A transformation format of ISO 10646, F. Yergeau, January 1998.

[B8] IETF RFC 2865, Remote Authentication Dial In User Service (RADIUS), C. Rigney, S. Willens, A. Rubens, W. Simpson, June 2000.

[B9] IETF RFC 3232, Assigned Numbers: RFC 1700 is Replaced by an On-line Database, J. Reynolds, January 2002.

[B10] IETF RFC 3279, Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, W. Polk, R. Housley, L. Bassham, April 2002.

[B11] IETF RFC 3411, An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, D. Harrington, R. Presuhn, B. Wijnen, December 2002.

[B12] IETF RFC 3412, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), J. Case, D. Harrington, R. Presuhn, B. Wijnen, December 2002.

[B13] IETF RFC 3413, Simple Network Management Protocol (SNMP) Applications, D. Levi, P. Meyer, B. Stewart, December 2002.

[B14] IETF RFC 3415, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), B. Wijnen, R. Presuhn, K. McCloghrie, December 2002.

[B15] IETF RFC 3416, Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP), R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, December 2002.

[18]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

[19]The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

[20]IETF RFCs are available from the Internet Engineering Task Force Web site at http://www.ietf.org/rfc.html.

[B16] IETF RFC 3417, Transport Mappings for the Simple Network Management Protocol (SNMP), R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, December 2002.

[B17] IETF RFC 3575, IANA Considerations for RADIUS, B. Aboba, July 2003.

[B18] IETF RFC 3576, Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS), M. Chiba, G. Dommety, M. Eklund, D. Mitton, B. Aboba, July 2003.

[B19] IETF RFC 3579, RADIUS Support for Extensible Authentication Protocol (EAP), B. Aboba, P. Calhoun, September 2003.

[B20] IETF RFC 3580, IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines, P. Congdon, September 2003.

[B21] IETF RFC 3748, Extensible Authentication Protocol (EAP), L. Blunk, J. Vollbrecht, B. Aboba, J. Carlson, H. Levkowetz, June 2004.

[B22] IETF RFC 4108, Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages, R. Housley, August 2005.

[B23] IETF RFC 4133, Entity MIB (Version 3), A. Bierman, K. McCloghrie, August 2005.

[B24] IETF RFC 4346, The Transport Layer Security (TLS) Protocol, Version 1.1, T Dierks, E. Resoria, April 2006.

[B25] IETF RFC 5216, The EAP-TLS Authentication protocol, D. Simon, B. Aboba, R. Hurst, March 2008.

[B26] ISO 6937-2:1983, Information processing—Coded character sets for text communication—Part 2: Latin alphabetic and non-alphabetic graphic characters.[21]

[B27] ISO/IEC 7498-1:1994, Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 1: The Basic Model.[22]

[B28] ISO/IEC 8824:1990, Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1) (provisionally retained edition).

[B29] ISO/IEC 8825:1990, Information technology—Open Systems Interconnection—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) (provisionally retained edition).

[B30] ISO/IEC TR 11802-2:1997, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Technical reports and guidelines—Part 2: Standard Group MAC addresses.

[B31] NIST FIPS 140-2, Security Requirements for Cryptographic Modules, 2002 December 3.[23]

[21]ISO publications are available from the ISO Central Secretariat, 1, ch. de la Voie-Creuse, Case Postale 56, CH-1211, Geneva 20, Switzerland (http://www.iso.org/). ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[22]ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[23]NIST publications are available from the National Institute of Standards and Technology, NIST Public Inquiries, NIST, 100 Bureau Drive, Stop 3460, Gaithersburg, MD, 20899-3460, USA (www.nist.gov).

[B32] NIST Special Publication 800-57, Recommendation for Key Management—Part 2: Best Practices for Key Management Organization, March 2007.

[B33] NIST Special Publication 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, E. Barker, J. Kelsey, June 2006.