IEEE STANDARDS ASSOCIATION

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

IEEE Std 802.1AR™-2018
(Revision of
IEEE Std 802.1AR-2009)

IEEE Std 802.1AR-2018
(Revision of
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 14 June 2018

IEEE-SA Standards Board

Abstract: A Secure Device Identifier (DevID) is cryptographically bound to a device and supports authentication of the device's identity. An Initial Device Identifier (IDevID) provide by the supplier of a device can be supplemented by Local Device Identifiers (LDevIDs) facilitating enrollment (provisioning of authentication and authorization credentials) by local network administrators.

Keywords: access control, authentication, authorization, certificate, IEEE 802.1AR, 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 978-1-5044-5019-5 STD23186
Print: ISBN 978-1-5044-5020-1 STDPD23186

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document 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

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents 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.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

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 Xplore at http://ieeexplore.ieee.org/ or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

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 by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent 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.

Essential Patent Claims may exist for which a Letter 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 completed, the IEEE 802.1 working group had the following membership:

Glenn Parsons, *Chair*
John Messenger, *Vice Chair*
Mick Seaman, *Security Task Group Chair, Editor*

SeoYoung Baek
Shenghua Bao
Jens Bierschenk
Steinar Bjornstad
Christian Boiger
Paul Bottorff
David Chen
Feng Chen
Weiying Cheng
Rodney Cummings
János Farkas
Norman Finn
Geoffrey Garner
Eric W. Gray
Craig Gunther
Marina Gutierrez
Stephen Haddock
Mark Hantel
Patrick Heffernan
Marc Holness
Lu Huang
Tony Jeffree
Michael Johas Teener
Hal Keen
Stephan Kehrer
Philippe Klein
Jouni Korhonen
Yizhou Li
Christophe Mangin
Tom McBeath
James McIntosh
Tero Mustala
Hiroki Nakano
Bob Noseworthy
Donald R. Pannell
Walter Pienciak
Michael Potts
Karen Randall
Maximilian Riegel
Dan Romascanu
Jessy V. Rouyer
Eero Ryytty
Soheil Samii
Behcet Sarikaya
Frank Schewe
Johannes Specht
Wilfried Steiner
Patricia Thaler
Paul Unbehagen
Hao Wang
Karl Weber
Brian Weis
Jordon Woods
Nader Zein
Helge Zinner
Juan Carlos Zuniga

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

Thomas Alexander
Johann Amsenga
Butch Anton
Stefan Aust
Christian Boiger
Paul Bottorff
Nancy Bravin
Vern Brethour
Demetrio Jr Bucaneg
William Byrd
Juan Carreon
Yesenia Cevallos
Keith Chow
Charles Cook
Sourav Dutta
Donald Eastlake, III
Janos Farkas
Andrew Fieldsend
Yukihiro Fujimoto
David Goodall
Eric W. Gray
Randall Groves
Michael Gundlach
Stephen Haddock
Marco Hernandez
Werner Hoelzl
Rita Horner
Russell Housley
Noriyuki Ikeuchi
Atsushi Ito
Raj Jain
SangKwon Jeong
Manabu Kagami
Piotr Karocki
Stuart Kerry
Yongbum Kim
Jeff Koftinoff
Hyeong Ho Lee
James Lepp
Jon Lewis
Michael Lynch
Elvis Maculuba
John Messenger
Michael Montemurro
Ronald Murias
Satoshi Obara
Thomas Palkert
Bansi Patel
Arumugam Paventhan
Clinton Powell
Karen Randall
R. K. Rannow
Alon Regev
James Reilly
Maximilian Riegel
Robert Robinson
Jessy Rouyer
Reinhard Schrage
Mick Seaman
Daniel Smith
Dorothy Stanley
Thomas Starai
Walter Struppler
Gerald Stueve
Mitsutoshi Sugawara
Bo Sun
Patrik Sundstrom
Mark-Rene Uchida
Dmitri Varsanofiev
George Vlantis
Khurram Waheed
Hao Wang
Lisa Ward
Karl Weber
Brian Weis
Andreas Wolf
Chun Yu Charles Wong
Dayin Xu
Yunsong Yang
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 14 June 2018, it had the following membership:

Jean-Philippe Faure, *Chair*
Gary Hoffman, *Vice-Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
Guido R. Hiertz
Christel Hunter
Joseph L. Koepfinger*
Thomas Koshy
Hung Ling
Dong Liu
Xiaohui Liu
Kevin Lu
Daleep Mohla
Andrew Myles
Paul Nikolich
Ronald C. Petersen
Annette D. Reilly
Robby Robson
Dorothy Stanley
Mehmet Ulema
Phil Wennblom
Philip Winston
Howard Wolfman
Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1AR-2018, IEEE Standard for Local and Metropolitan Area Networks—Secure Device Identity.

This standard specifies Secure Device Identifiers (DevIDs) for use with IEEE Std 802.1X™ [B1][1] and other industry standards and protocols that authenticate, provision, and authorize communicating devices.

Each DevID comprises an RFC 5280 conformant X.509 certificate that identifies the subject device and can include authorization information signed by the certificate's issuer, a secret private key that corresponds to the certificate's subject public key, and any certificate chain required to facilitate the certificate's use. A device's DevID module stores each of its DevID secrets securely and supports signing operations that prove possession of the secret (and thus that the device is the subject of the associated DevID certificate), while ensuring that the secret remains confidential so the device cannot be impersonated by others.

An Initial Device Identifier (IDevID) provided by a device's supplier can be supplemented by one or more Local Device Identifiers (LDevIDs), each using an existing or a freshly generated secret, facilitating enrollment (provisioning of authentication and authorization credentials to authenticated devices) by a local network administrator.

The first edition of IEEE Std 802.1AR was published in 2009. This revision added the ECDSA P-384/SHA-384 signature suite option; removed the RSA-2048/OPAQUE option (that permitted the use of an undisclosed hash function); restructured the document to enable future signature suite changes, for clarity (particularly in conformance statements and the PICS), and revised the MIB. A DevID module can now implement more than one signature suite (facilitating interoperability and the use of a device in different authentication environments) and additional service operations (that do not conflict with mandatory requirements) as long as these are disclosed (facilitating backwards compatibility and support of DevID functionality by other modules, e.g., TPM).

[1]Numbers in brackets correspond to entries in the Bibliography in Annex C.

Contents

Figures

Tables

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. Moreover, 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 can result in a “chicken-and-egg” scenario, wherein these credentials 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 [B4]) and other industry standard authentication and provisioning protocols.[1] A standardized device identity facilitates interoperable secure device authentication that helps simplify and standardize secure deployment and management of devices. A device is any entity in an IEEE 802 LAN that seeks to obtain services from the network or provide services on the network.

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

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 can 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.

Multiple logical or physical devices, each with its own unique DevID can be contained within an aggregate device. The selection of a DevID for the aggregate device can depend on the context in which it is to be identified. This standard assumes that any such selection has been made and addresses device requirements independent of their simple or aggregate nature.

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. This facilitates secure device provisioning.

1.3 Relationship to other standards

This standard specifies 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, it attempts compatibility with the following:

a) Trusted Platform Module (TPM)

NOTE—TPM Keys for Platform Identity [B13] describes how TPM 1.2 can be used to provide DevID functionality, superseding IEEE Std 802.1AR-2009 Annex B.

b) Extensible Authentication Protocol-Transport Layer Security (EAP-TLS [B6])

IETF RFC 7030 [B9] (Enrollment over Secure Transport) describes a certificate management protocol for Public Key Infrastructure (PKI) clients that need to acquire client certificates and associated Certification Authority (CA) certificates. A client can use an IDevID, as defined by this standard, to participate in the enrollment protocol which supports both client generated and CA generated public/private key pairs (LDevIDs).

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.

IEEE Std 802.1AC™, IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Service Definition.[2, 3]

ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).[4]

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.[5]

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 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 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 4055, Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Schaad, J., Kaliski, B., Housley, R., June 2005.

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

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.

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.

IETF RFC 6353, Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP), Hardaker, W., July 2011.

---

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

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

[4]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/).

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

IETF RFC 6933, Entity MIB (Version 4), Bierman, A., Romascanu, D., Quittek, J., Chandramouli, M., May 2013.

IETF RFC 8017, PKCS #1: RSA Cryptography Specifications Version 2.2, Moriarty, K., Kaliski, B., Jonnson, J., Rusch, A., November 2016.

ISO/IEC 8825-1 Information technology—ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).[6]

NIST FIPS 180-4, Secure Hash Standard (SHS), August 2015.[7]

NIST FIPS 186-4, Digital Signature Standard (DSS), July 2013.

NIST Special Publication 800-90A, Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, E. Barker, J. Kelsey, June 2015.

[6]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, CO 80112, USA (http://global.ihs.com/).

[7]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).

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[8]

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

3.2 authentication: Verification that a claimed identity is correct.

3.3 authentication exchange: The information exchange between entities performing authentication.

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

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

3.5 Basic Encoding Rules (BER): Rules, specified in ISO/IEC 8825-1, for encoding ASN.1 data types.

3.6 certificate: A digitally signed object that binds information identifying an entity that possesses a secret private key to the corresponding public key.

3.7 certificate chain: An ordered list of intermediate certificates (3.30) that links an end entity certificate (in this standard, a DevID certificate) to a trust anchor.

3.8 certificate revocation list (CRL): A signed list of revoked certificates.

3.9 certificate signing request (CSR): A signed message from the device to the CA requesting a certificate be issued.

3.10 certification authority (CA): An entity that issues X.509 digital certificates.

3.11 cipher suite: A set of one or more cryptographic algorithms.

3.12 client: A protocol entity that makes use of a service.

3.13 credential: Information that an entity (a person or device) possesses that allow it to make a verifiable claim of identity, i.e., to be authenticated.

NOTE—In this standard, DevIDs are credentials.

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

3.15 cryptographic key: An input parameter that varies the result of applying a cryptographic function.

3.16 customer: In this standard, the person, organization, or administrator acting on their behalf, that allows a device received from its supplier to be attached to and subsequently used in the customer's network.

3.17 device: A device is any entity that has an IDevID (3.25).

[8] *IEEE Standards Dictionary Online* is available at https://ieeexplore.ieee.org/xpls/dictionary.jsp

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

3.18 DevID: A device identifier that is cryptographically bound to the device, and comprises a DevID secret (3.21), a signed DevID certificate (3.19) that binds possession of that secret to a statement of identity made by the certificate's issuer, and (as required by authenticating systems) a certificate chain (3.7) that links the certificate to a trust anchor (3.43).

3.19 DevID certificate: A data object constructed using cryptographic operations to bind the DevID Name and other data to a DevID secret (3.21) possessed by the device.

3.20 DevID module: A logical security component that securely stores and operates on DevID secret(s) and associated DevID certificate(s).

3.21 DevID secret: The private key portion of a public-private key pair bound to a DevID certificate.

3.22 DevID solution: The systems, protocols, and/or the policies and procedures that support the use of DevID equipped devices in a customer network.

3.23 DevID trust anchor store: The database of trust anchor information for IDevIDs and LDevIDs that is stored and used by a DevID solution. This is equivalent to the common Web browser trust anchor store and can be shipped with the DevID solution (3.22).

3.24 Distinguished Encoding Rules (DER): A subset, specified in ISO/IEC 8825-1, of the Basic Encoding Rules (BER, 3.5) that specifies exactly one way of encoding any particular ASN.1 value.

3.25 IDevID: A DevID (3.18) installed in a DevID module (3.20) by the supplier of the device.

3.26 enrollment: The process and protocols used by customer network systems to recognize a device possessing an IDevID (3.25) and authorize it for subsequent network activity, possibly including giving it an LDevID (3.32).

3.27 fingerprinting: The process of uniquely identifying (with sufficiently high probability) a personal device or person using a device by observing the network traffic sent and received by the device.

3.28 IEEE 802 Local Area Network (LAN): IEEE 802 LANs are LAN technologies that provide a MAC Service equivalent to the MAC Service defined in IEEE Std 802.1AC. IEEE 802 LANs include IEEE 802.3™ (CSMA/CD) and IEEE 802.11™ (wireless).

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

3.29 Initial Secure Device Identifier: *See:* **IDevID** (3.25).

3.30 intermediate certificate: A certificate that is part of the certificate chain (3.7) stored by the DevID module (3.20) for a DevID (3.18).

3.31 key: *See:* **cryptographic key** (3.15).

3.32 LDevID: A locally significant DevID (3.18) that is unique in the administrative domain in which the device is used.

3.33 local entity: A protocol entity that makes use of a DevID module's service interface or management interface and is implemented in the same device as that module.

3.34 Locally Significant Secure Device Identifier (LDevID): *See:* **LDevID** (3.32).

3.35 personal device: A device used by an individual or a small group of people, such that identification of the device or its network activity implies the location or activity of that individual or a group member.

3.36 Public Key Infrastructure (PKI): A set of network entities and the roles, policies, and procedures that govern the creation, distribution, use, storage, and revocation of X.509 digital certificates.

3.37 PKI hierarchy: A relationship between systems supporting a PKI, where systems with a role associated with a tier in the hierarchy can delegate authority to a system or systems whose role is associated with an immediately lower tier.

3.38 Registration Authority (RA): A PKI entity that verifies requests for X.509 digital certificates before requesting their issuance by a CA.

3.39 Secure Device Identifier: *See:* **DevID** (3.18).

3.40 Secure Device Identifier Module: *See:* **DevID Module** (3.20).

3.41 signature suite: An asymmetric cryptographic algorithm and associated constraints (e.g., restrictions on key types, key sizes, and component functions) that facilitate its implementation and use to generate digital signatures with specific security properties.

3.42 supplier: In this standard, the person, organization, or administrator acting on their behalf, making a claim of conformance in respect of a device that is provided for attachment to a customer network.

3.43 trust anchor: A CA (3.10) that is trusted and for which the trusting party holds information, usually in the form of a self-signed certificate issued by the trust anchor.

3.44 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.

4. Acronyms and abbreviations

ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CA	Certification Authority NOTE—CA is also an abbreviation for secure Connectivity Association in IEEE Std 802.1AE™ and IEEE Std 802.1X™.
CP	Certificate Policy
CPS	Certification Practices Statement
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DN	Distinguished Name
DRBG	Deterministic Random Bit Generator
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
PKI	Public Key Infrastructure
RA	Registration Authority
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

SP	Special Publication
TLS	Transport Layer Security
TPM	Trusted Platform Module

5. Conformance

A claim of conformance to this standard is a claim that a device meets the requirements of this standard as they apply to the use, format, secure storage, and management of Secure Device Identifiers (DevIDs).

Conformance to this standard does not ensure that a device 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 *cannot*. The use of *needs to* highlights conditions that have to be met for successful use of this standard but are not a direct consequence of a conformant implementation.

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 support one or more of the signature suites specified in Clause 9 with a DevID module that:

a) Contains an IDevID, as specified in Clause 6, for each supported signature suite.

b) Stores DevID secrets as specified in 7.1.

c) Supports the mandatory service interface operations specified in 7.2 a) through g).

d) Includes all fields as required by Clause 8 in IDevID certificates and IDevID intermediate certificates.

e) Has an X.500 DN in the `subject` field of each IDevID certificate (8.6).

f) Restricts Simple Network Management Protocol (SNMP) access to DevID management capabilities, if provided, to the use of SNMP version 3 in privacy/authentication mode or SNMP over TLS as specified in RFC 6353.

If the DevID module supports key generation (7.2.8) or generates random numbers for use by protocol entities outside the DevID module, it shall:

g) Generate random numbers that are unpredictable and meet or exceed the entropy requirements for creating keys of the strength required for each supported signature suite, as specified in 7.1.3 and Clause 9.

A device for which conformance to this standard is claimed shall not:

h) Permit remote access to the signing operations provided by the DevID module (7.2.5).

i) Include any field or extension specified as prohibited in an IDevID certificate or intermediate certificate (8.10).

5.4 Optional capabilities

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

a) Support the use of at least one LDevID and the associated mandatory and optional operations specified in 7.2.

b) Include a unique device `serialNumber` in the `subject` field of each IDevID certificate (8.6).

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

c) Perform cryptographic zeroization on LDevID keys, certificates, or certificates when supporting delete operations (7.2.10, 7.2.13, 7.2.14).

d) Use a non-deterministic or deterministic RNG (7.1.3) and if using a deterministic RNG should:

   1) Use one of the DRBGs listed in SP 800-90A Revision 1 (7.1.3).

   2) Support the addition of RNG entropy (7.2.15).

e) Support DevID management operations (7.3) using SNMP and the DevID MIB (Clause 10).

f) Include the `subjectAltName` in each IDevID certificate (8.10.4).

A device that includes the `subjectAltName` in each IDevID certificate should:

g) Include a `HardwareModuleName` in the `subjectAltName` (8.10.4).

5.5 Supplier information

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

a) Publish an RFC 3647 compliant Certificate Policy (CP) and Certification Practices Statement (CPS) covering the practices used by the supplier's CA to issue, manage, and revoke IDevID certificates.

b) Make available trust anchor information for each IDevID as specified in 6.3, and 6.4, in the form of a self-signed certificate as specified by RFC 5280.

and shall disclose the following information in the PICS:[10]

c) The IDevID subject name fields that can be used for authorization, auditing, or other purposes and how they are parsed (Clause 6).

d) How IDevID secrets (private keys) are generated and installed in the DevID module (6.1).

e) Plans to deal with any compromise to the supplier's CA signing credentials or IDevID secrets (6.4).

f) How the DevID module preserves the confidentiality of DevID secrets (7.1).

[10]This is done, as stated in the PICS' instructions, by including a reference in the PICS proforma and providing the referenced information, possibly in documents provided to meet other disclosure requirements.

g) Any restrictions on the integrity and availability of DevID module functionality (7.1).

h) Any operations supported by the DevID module in addition to those specified in 7.2.

i) The basis of the supplier's belief in the uniqueness of the `issuer` field (8.4) in the IDevID certificates, intermediate certificates, and self-signed certificates for trust anchors.

NOTE—The security inherent in the systems and practices used to issue and manage certificates and to generate and install keys might be reflected in any certifications pursued for the devices in question, for example FIPS 140-2 [B11].

6. Secure Device Identifiers (DevIDs) and their use

This clause describes DevIDs and their use. It provides the context necessary for understanding the DevID module and its operation, as specified in Clause 7.

A DevID comprises:

a) A DevID secret (6.1) that is the private key portion of a public-private key pair;

b) A DevID certificate (6.2) containing the corresponding public key and a subject name that identifies the device; and

c) The certificate chain (6.3) from the DevID certificate up to a trust anchor contained in the DevID trust anchor store (see 6.4) available to potential authenticators.

NOTE—If the DevID trust anchor store includes the certificate of the CA that signed the DevID certificate, there need not be an explicit certificate chain on the device.

Possession of a DevID allows a network attached device to assert its identity in authentication protocols. One or both of the network entities participating in an authentication exchange might use a DevID. DevID based authentication and authorization solutions can use access controls that include (for example) lists of the device types, device serial numbers, or other attributes that can be included in the subject name fields (8.6) of DevID certificates. The contents of those fields are cryptographically bound to the certificate, which is validated and access controls applied before allowing the device to take some action (e.g., connecting to the network).

A device proves the identity associated with a DevID certificate by performing a signing (7.2.5) operation that uses the DevID secret. One or more DevIDs can make use of the same secret and the corresponding public key, as more than one DevID certificate can include a given public key. The DevID module (Clause 7) stores and uses each DevID secret within a controlled boundary (the *cryptographic boundary*), limiting its use to specific service interface operations (7.2) and preventing exfiltration. Access to service interface operations is further limited to those components of device operation that can be trusted to participate in the exchange of authentication and authorization protocol information, and not (for example) to act as a proxy for another device wishing to access the signing operation for the purpose of impersonating the device.

DevIDs are explicitly created. IDevIDs (Initial Device Identifiers) are created before the device is supplied to the customer. A DevID module includes an IDevID for each of the signature suites (Clause 9) that it supports. An LDevID (Locally Significant Device Identifier) is typically created when a device is first attached to a customer network, but can be created at any time, as permitted by the device administrator. An LDevID can make use of an existing IDevID secret; alternatively, the DevID module can generate a fresh LDevID public and private (DevID secret) key pair internally (7.2.8), or can securely store an externally generated secret (7.2.9) transferred to the device. LDevID certificates and certificate chains are also stored by the DevID module (7.2.11, 7.2.12). LDevID certificate fields can reflect authorization decisions based both on the fields in authenticated IDevID certificates and other information provided by the device's supplier or user.

DevIDs are used and maintained in accordance with the policies established by network and device administrators. The DevID module service interface (7.2) provides operations to list (7.2.2, 7.2.3, 7.2.4), enable and disable (7.2.6, 7.2.7), and delete DevID components (7.2.10, 7.2.13, 7.2.14). Each DevID can continue to exist for as long as its DevID module remains operational.

DevID creation implies the use of additional systems and protocols within a network to obtain signed DevID certificates. While the detailed specification of these is outside the scope of this standard, the trust that can be placed in the resulting DevIDs is of prime concern when choosing options provided by this standard and selecting infrastructure components. The PKI hierarchy and trust model is summarized in 6.4 and privacy considerations in 6.5.

Once a device has been enrolled in a customer network it can use a DevID for authentication and as an input to authorization decisions. Annex B provides additional examples and considerations.

6.1 DevID secrets

The DevID secret for a DevID supported by a signature suite (Clause 9) specified in this standard is a private key as defined and used by that signature suite.

6.1.1 DevID secret creation

The DevID module can use an internal random number generator (7.1.3) to generate DevID secrets, or they can be generated externally and installed in the device.

The quality of a DevID secret greatly influences the quality of any DevID certificate that makes use of that secret. Poor randomness during internal generation and inappropriate management of external generation can both compromise security. Clause 9 specifies key strength requirements. Care needs to be taken to clear any externally held residual key material for a DevID secret that has been inserted into the DevID module.

6.1.2 DevID secret insertion

The capability to insert LDevID secrets into the DevID module needs to be limited to those components of device operation that can be trusted to protect the confidentiality and integrity of each LDevID secret as it is conveyed between the system generating the LDevID secret and the DevID module, and to authenticate and authorize that generating system. If the DevID module is not capable of internal key generation, special care has to be taken when installing the IDevID secret to avoid the acquisition of that secret by an entity that might interpose itself between the generator of that secret and the target device, as no existing secret is available.

NOTE—Internal generation of DevID secrets is recommended (see 7.2, 7.2.8).This standard mandates the disclosure of IDevID key generation and certificate signing mechanisms (5.5, A.9) by the supplier of the device.

6.2 DevID certificates

DevID certificates are a subset of the X.509 v3 certificates profiled in RFC 5280, as specified in Clause 8.

Every DevID certificate needs to be unique among certificates issued by a given CA, while the CA itself needs to be uniquely identified (by DevID certificate fields, as specified in 8.4) among those CAs that sign DevID certificates for a particular domain of device use. DevID certificates, issued by a given CA for distinct devices, require a unique subject name as described in 8.6 (`subject`) and 8.10.4 (`subjectAltName`). DevID certificates issued by the same CA with the same `subject` and `subjectAltName` but with a different `serialNumber` (8.2) are considered to identify the same device.

NOTE—In this clause, this `font` identifies DER encoded fields in certificates.

6.2.1 DevID certificate creation

Creation of a DevID certificate requires knowledge of the public key corresponding to the DevID secret to be used with that certificate and knowledge of the attribute values to be included in the device's subject name, as well as possession of credentials necessary to sign the certificate or (in the case of an RA) trusted access to a CA that will sign the certificate.

If the device already possesses the DevID secret it can generate and sign (7.2.5) a Certificate Signing Request (CSR) with a proposed subject name, using the subject name from an existing DevID certificate or using other device attributes such as a machine readable device serial number. The RA or CA can then

validate the CSR, construct the DevID certificate, and return it to the device for insertion (7.2.11) into the DevID module. Alternatively, an RA can extract the required subject name fields and the public key from a DevID certificate used by the device with TLS to secure a connection to the RA, returning and inserting the new DevID certificate without requiring a prior CSR.

However obtained, the contents of a new DevID certificate's subject name and other fields are not limited to information provided by the device. In an extreme example, a succession of identical devices are physically connected (securely, and one at a time) to an RA that inserts both an IDevID secret and an IDevID certificate using that secret, thus conferring a unique identifier on each device. Note, however, that this manufacturing example requires operations beyond the scope of those specified in Clause 7. The latter supports only the secure insertion of LDevID secrets into DevID modules that already possess an IDevID, not the unprotected insertion of IDevID secrets.

In another example, a DevID solution controlling the use of a device in a customer network bases its decision to create an LDevID certificate and the fields in that certificate on the following:

a) Information from an IDevID certificate presented by the device;

b) Information supplied by the device's operator (making use of the IDevID to secure a connection to the RA); and

c) Information provided to the customer network administrator by the device's supplier.

The DevID certificate is used to identify the supplier of the device, the device type, and the device serial number. The information provided by the device supplier to the network administrator is used to confirm that a device of that type and with that serial number has been shipped to that operator (and not diverted from its intended destination) and to supply any other information about the configuration of the device that is not included in the IDevID but required by the customer. The information supplied by the device operator is used to confirm the operator's identity and to confirm or request authorization for specific network services. Fields in the created LDevID certificate then reflect explicit authorization granted to the device and might also identify its operator (by customer network account number, for example). This example also illustrates how DevID use can facilitate direct shipping of a device by a supplier to its final location or operator, without having to be first staged at a customer network provisioning center.

NOTE—These examples are not an authoritative guide to enrollment (the process by which customer network systems recognize a device and authorize it for subsequent network activity) but have been provided here (rather than in an informative Annex) to provide the reader with the necessary context to understand the range of capabilities inherent in the use of DevIDs and DevID modules prior to reading Clause 7.

6.2.2 DevID certificate validation

Validation of a DevID certificate confirms the authenticity of the device presenting the DevID certificate and possessing the corresponding DevID secret (i.e., confirms that the device is what is claimed in the certificate, which includes its subject name). The DevID certificate is an X.509 certificate and network entities participating in an authentication exchange with the device can validate it using the RFC 5280 defined mechanisms. The validation process requires verification of signatures generated by the DevID's signature suite (Clause 9) and the decoding of X.509 certificates using the distinguished encoding rules, as described in RFC 5280.

IDevIDs are intended to have very long validity periods (8.5) that do not constrain device lifetimes and exceed those of certificates that are periodically renewed. DevID validation can include Certificate Revocation List (CRL) checking, though the supplier of a device is not required to provide a CRL.

An authenticator that validates a DevID certificate needs to have (as described in 6.4, trust anchor information for the CA that signed the last of a chain of certificates that begins with the CA that signed the DevID certificate. The DevID module itself stores the certificate chain (6.3) so that it can be presented to

authenticators. This standard does not specify protocols that exchange information between DevID modules and potential authenticators. Use of the TLS authentication protocol is discussed in B.1.

6.3 DevID certificate chains

The DevID module needs to include the entire certificate chain for each DevID up to the trust anchor; inclusion of the self-signed certificate for the trust anchor is also recommended. For a shallow hierarchy, in which the DevID has been signed by the trust anchor, this requirement could be met without including an explicit certificate chain on the device. The supplier of the device needs to make the IDevID trust anchor information available in a way that is convenient for, and can be trusted by, customer network administrators and DevID solution providers.

6.4 DevID Trust Model

Figure 6-1 illustrates the DevID trust hierarchy. IDevID certificates are typically issued and signed by a manufacturer's CA, though the manufacturer might simply provide an RA (Registration Authority) and use a CA administered by a third-party or the eventual supplier of the device if that is a different organization. The CA that signs the IDevID is typically part of a larger hierarchy; this standard places no requirements on its depth and there is no global trust anchor. Use of an LDevID implies the existence of a local CA. This might use a complex PKI hierarchy with RAs, or a lower cost infrastructure instantiated by a single local CA.

NOTE—As specified in Clause 5, a conformance claim is made by the device's supplier.

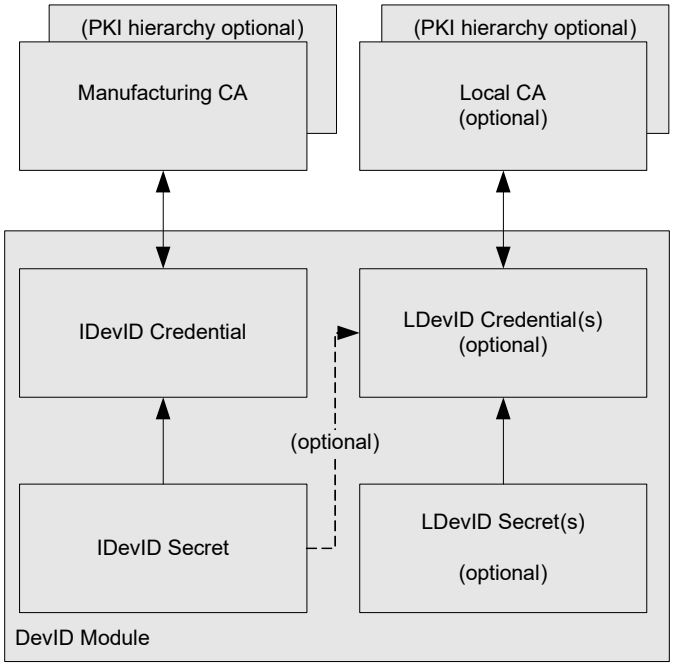

Figure 6-1—DevID trust hierarchy

DevID based authentication and authorization solutions validate DevIDs using a DevID trust anchor store. Local configuration of this store allows the network administrator to add or remove trust anchor information for IDevIDs (provided by the device suppliers) or LDevIDs.

Risks inherent in this trust model include the following possibilities:

a) The supplier's signing credentials are compromised: directly by disclosure of a private key used by the supplier's trust anchor or a CA intermediate in its PKI hierarchy between the trust anchor and signed IDevIDs; or, one of those CAs has been tricked or coerced into signing a certificate for a CA controlled by an attacker.

b) The DevID based authentication solution's trust anchor store includes incorrect entries: injected into the list maintained by the supplier; or acquired by the customer as a result of being misdirected from the supplier to an alternate list.

c) A device's DevID secret has been disclosed.

If a supplier's signing credentials or PKI hierarchy have been compromised [item a) and item b) above], an attacker could create a fake IDevID certificate. A customer can install an LDevID certificate to limit exposure to compromising scenarios that occurred prior to initial device deployment. The supplier is expected to use the same mechanisms that they would use for any other high-priority security vulnerability found in their product line when announcing potentially compromised signing credentials. DevID solutions are expected to allow local configuration (subject to authentication and authorization of the customer network administrator) of trust anchor stores, so unwanted certificates can be removed and new ones added. The administrator is expected to examine the CP published for the device manufacturer's CA and the PICS proforma for the DevID-compliant devices before adding supplier trust anchor information. Validation of device types and serial numbers (or other DevID certificate subject name fields) against lists of devices expected from suppliers can help prevent authorization of rogue devices.

If an attacker acquires a device's DevID secret [item c) above], that attacker can clone the device. The clone can spoof the original device, even if it has a different DevID certificate subject name, because it can perform signing operations using the same secret key. Use of a one-time authorization mechanism that is tied to LDevID provisioning and that records the public keys of previously authorized devices can prevent a cloned device from subsequently joining the network.

If the original manufacturer and the supplier of the device are separate organizations, the latter might wish to substitute its own IDevID in much the same way as an LDevID is substituted for an IDevID, but using additional DevID module operations (not specified in Clause 7) to remove the prior IDevID and reduce its exposure to the risks described in this clause.

NOTE—Bootstrapping Remote Security Key Infrastructures (BRSKI) [B14], under development at the same time as the 2018 revision of this standard, specifies protocols and a framework for automated enrollment information exchange between supplier and customer network systems.

6.5 Privacy considerations

A DevID could be used to track a personal device, potentially violating the privacy of its owner or operator. This threat to privacy can be reduced by the selective and careful use of DevIDs in general, and of LDevIDs in particular. For example, the use of peer identity privacy provided by the authentication protocol TEAP specified in RFC 7170 [B10] allows the personal device to delay revealing its identity until a TLS tunnel has been established and it has had the opportunity to authenticate the network to which it is to be attached. A device that might be attached to a number of networks can then be authenticated by each without revealing (to that network as well as to casual eavesdroppers) its use of other networks, and the identities associated with those networks.

Maintaining privacy requires attention to the use of and potential commonality between DevID certificate fields. For example, if a number of LDevIDs share a common secret (possibly the IDevID secret) the associated `subjectPublicKeyInfo` effectively fingerprints the device.

The operations specified in Clause 7 allow the device administrator to selectively enable or disable the use of IDevIDs and LDevIDs. A management agent that is part of the device can also restrict remote management access, depending on the identity of the manager seeking access, to particular DevIDs.

7. DevID Modules

This clause describes the functionality provided by a DevID Module. Clause 6 describes DevIDs and their use. The details of DevID certificates and intermediate certificate fields are specified in Clause 8.

7.1 DevID module functionality

A DevID module is a logical security component that stores and operates on the DevID(s) associated with a device. Figure 7-1 illustrates the mandatory and optional functionality provided by the module including: storage (7.1.1) for one or more DevIDs; asymmetric cryptography (7.1.2) for signing, decryption, and integrity checking with a DevID secret; and random number generation (7.1.3) for DevID secret (private key) generation. Each DevID module supports one or more of the signature suites specified in Clause 9.

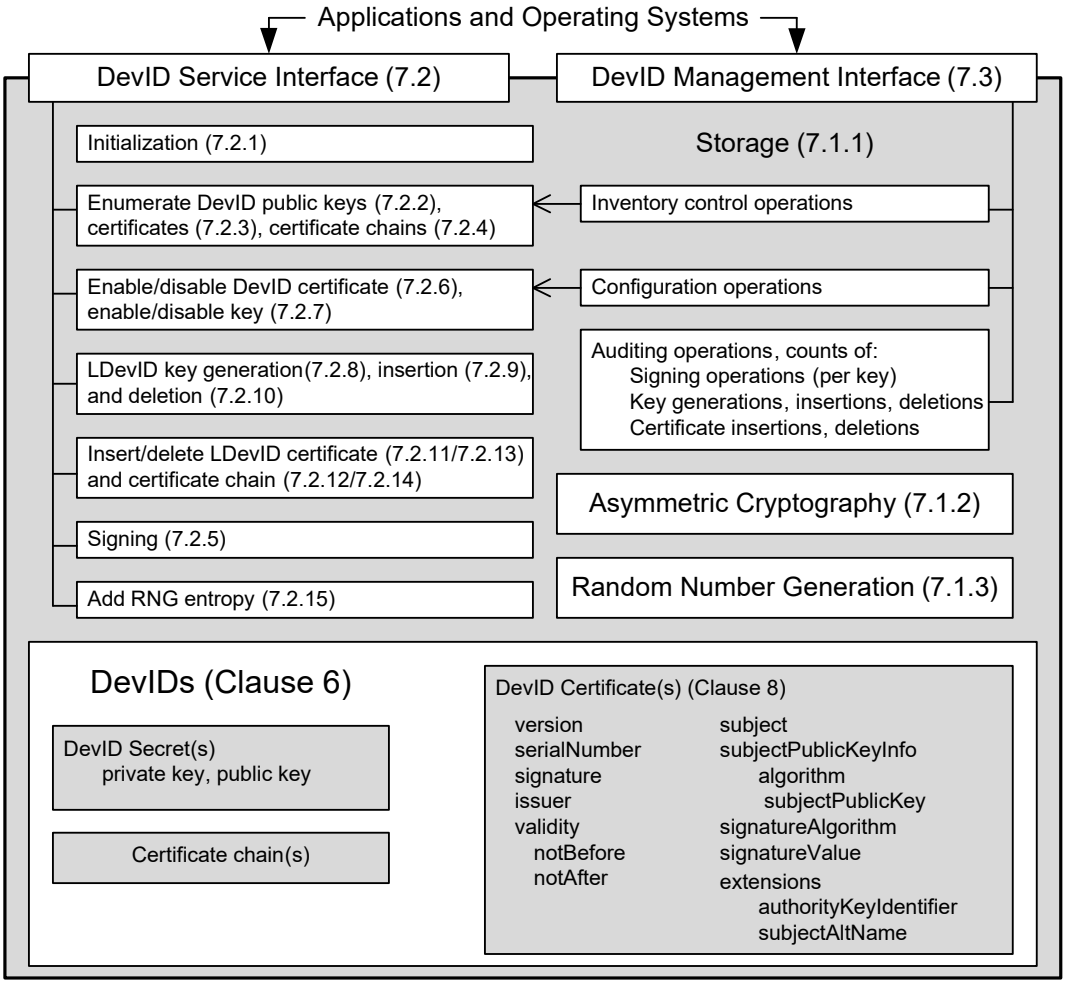

Figure 7-1—DevID functionality

DevID Service Interface (7.2) and DevID Management Interface (7.3) operations provide access to DevID module functionality without disclosing DevID secrets, which are stored and used within a controlled boundary (the *cryptographic boundary*). It is imperative that DevID secrets are securely protected from access by any entity outside the cryptographic boundary. External disclosure would result in a compromised DevID that would no longer be bound to the device. DevID certificates are available to any entity that needs to verify the identity of a device.

7.1.1 Storage

The DevID module is responsible for maintaining the integrity of its internal state, and ensuring the availability of service interface operations whenever the device might need to assert its identity. In many implementations that means that those operations need to have the same availability as boot code, and that internal data is integrity protected against inconsistencies arising from power failures during update.

The DevID module includes an IDevID (Clause 6) for each of the signature suites (Clause 9) supported by the DevID module and zero or more LDevIDs installed as a result of enrolling the device using the IDevID to prove authenticity and protect the installation process (6.2.1). DevID secrets shall be stored confidentially by the DevID module, shall be accessed only through the operations provided by the DevID Service Interface, and shall not be retrievable from the module.

NOTE 1—A DevID secret module might provide storage for DevID secrets in any one of a number of different ways. These include hardware secured storage within the module (e.g., embedded one-time programmable memory), encrypted storage external to the module, and the use of a nominally unsecured storage location where the device's operating system restricts all access (software secured storage).

Example storage sizes for the DevID secret, as maintained in various formats, are listed in Table 7-1.

Table 7-1—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)
ECPrivateKey	namedCurve as described in RFC 5480 and RFC 5915 [B8].	119 (for a prime256v1 key)	The following are included in an ECPrivateKey: version privateKey (OCTET STRING) parameters publicKey (BIT STRING)

NOTE 2—For both RSA and ECC private-public key pairs, the possession of a private key implies possession of the corresponding public key as the latter can be readily calculated from the former (though naturally not vice-versa). So, while the requirement for the DevID module to store a DevID secret confidentially is emphasized, service interface operations (7.2) can return the corresponding public key (whether explicitly stored internally or not) or use it to reference the secret, and a reference to the stored secret key is sufficient to identify the key pair.

7.1.2 Asymmetric cryptography

Asymmetric cryptography uses pairs of keys, each pair comprising a public key that can be shared freely and a private key. Derivation of the private key from the public key is computationally infeasible, at least to the degree associated with the strength inherent in the asymmetric cryptographic algorithm and the size and types of the keys. Knowledge of the public key allows a digital signature computed using the private key to be verified, proving that the entity producing the signature possessed the private key.

The DevID module supports one or more signature suites (Clause 9). These specify cryptographic algorithms used for signing (7.2.5) characterized with the values of parameters (for example, key types and sizes) used by those algorithms, and their representation in DER encoded certificate fields.

7.1.3 Random number generation

The DevID module provides a random number generator that meets the security strength requirements (as specified in NIST SP 800-90A Revision 1 and Clause 9 of this standard) for each signature suite for which it supports key generation (7.2.8). If a non-deterministic RNG (e.g., hardware RNG) is not available, the system makes use of sufficient entropy to create a seed of the required quality for a deterministic RNG and the RNG should use one of the DRBGs listed in NIST SP 800-90A Revision 1. RFC 4086 [B5] provides a self-described “best current practice” document for producing random numbers. The DevID module service interface supports the addition of entropy (7.2.15) at any time. When sufficient entropy is not used to create a seed of the required quality, generated keys and nonces can be ineffective against published attacks (see, for example, Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices [B15]). It is imperative that the internal state of that RNG be secure against access by any entity outside the cryptographic boundary, but the DevID module can also generate random numbers for use by protocol entities operating outside that boundary.

7.2 DevID Service Interface

DevID module operations provide direct control over the device’s identity. Access to all these operations, with the exception of initialization and signing, can be made available via a local entity to a local or remote administrator (via remote management protocol) but needs to be limited as appropriate for the device and administration model. See 6.5 for privacy considerations. Details of other security considerations and resulting implementations are out of the scope of this standard. These operations make the DevID module useful for assertion of the DevID identity. Examples of their use are found in Annex B.

Each of the operations implemented by a DevID module shall be capable of supporting each of the signature suites supported by that module [5.3 c)].

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

a) Initialization (7.2.1)
b) Enumeration of the DevID public keys (7.2.2)
c) Enumeration of DevID certificates (7.2.3)
d) Enumeration of DevID certificate chains (7.2.4)
e) Signing (7.2.5)
f) DevID certificate enable/disable (7.2.6)
g) DevID key enable/disable (7.2.7)

A DevID module that supports the use of LDevIDs should implement one or both of the following operations:

h) LDevID Key generation (7.2.8)
i) LDevID Key insertion (7.2.9)

A DevID module that implements LDevID Key generation or insertion shall implement the following operations:

j) LDevID Key delete (7.2.10)

A DevID module that supports the use of LDevIDs shall implement the following operations:

k) LDevID certificate insert (7.2.11)

l) LDevID certificate chain insert (7.2.12)

m) LDevID certificate delete (7.2.13)

n) LDevID certificate chain delete (7.2.14)

The DevID module that makes use of a deterministic RNG should implement the following operation:

o) Addition of RNG entropy (7.2.15)

A DevID module may implement additional service interface operations provided that they do not facilitate exfiltration of a DevID secret or signing of data not under the control of a local trusted protocol entity. These additional operations can include deletion of the IDevID certificate or IDevID key, which can be logically equivalent to decommissioning the device, and shall be disclosed by the supplier of the device.

NOTE—Additional service operations meeting these constraints are explicitly permitted to allow DevID functionality to be provided by modules with additional capabilities (such as a TPM) or other editions of this standard.

This subclause (7.2) specifies the functionality to be provided by each of the service interface operations but does not otherwise constrain implementation details. In particular it is not an application program interface in any particular computer language. In this specification it is convenient to use a keyType to identify the signature suite and a keyIndex or certificateIndex to refer to each key pair or certificate stored by the DevID module. These types and indices are of purely local significance and can be implemented in any way convenient for a particular module or device. It is up to that implementation to handle any issues that might arise from local changes to, or reuse of, index values. Remote management operations (7.3) can use a fingerprint of a certificate to refer to that certificate and a fingerprint of the `subjectPublicKeyInfo` to refer to a public key or its corresponding secret key (10.3).

7.2.1 Initialization

Input: None

Output: Availability or unavailability of other DevID Service Interface operations.

This operation causes the DevID module to prepare itself for use. For example, any self-test operations necessary to comply with FIPS 140-2 [B11] are performed.

This operation is not expected to be exposed to the device administrator but is used by the OS or firmware to initialize the DevID module. The other DevID service operations are not available until initialization completes and will continue to be unavailable if the module's internal consistency checks fail, if it detects a compromise of its protected data, or if access has been otherwise disabled.

7.2.2 DevID public key enumeration

Input: None

Output: A table containing, for each key pair:

— The keyIndex,

— A value indicating if the key is enabled,

— The `subjectPublicKeyInfo` for that key as specified for the signature suite in Clause 9, and

— A value indicating if the key is used by the IDevID certificate for that signature suite.

The keying material returned is limited to the public key. The DevID module is responsible for the integrity of stored keys (7.1.1) and does not report any previously installed key whose integrity has been detected as compromised. The DevID management interface (7.3) records integrity errors detected when service interface operations are performed and at other times.

7.2.3 DevID certificate enumeration

Input: None

Output: A table containing, for each certificate:

- The certificateIndex,
- The associated keyIndex,
- A value indicating if the certificate is enabled,
- A value indicating if the certificate is an IDevID certificate, and
- The certificate itself, DER encoded as specified in RFC 5280.

If a DevID certificate's key has been deleted, as a result of an explicit administrative operation (7.2.10), internal failure, or integrity check, a null value will be returned instead of the keyIndex and the DevID will no longer be usable. The service operations specified in this clause do not provide a way to associate a DevID certificate stored by the module with a subsequently inserted key, but the administrator can attempt to delete the certificate and reinsert both key and certificate if appropriate.

7.2.4 DevID certificate chain enumeration

Input: The certificateIndex of the DevID certificate for which the certificate chain is requested.

Output: A table containing each certificate in the chain, DER encoded as specified in RFC 5280.

The trust anchor information can be part of the chain.

7.2.5 Signing

Input: The keyIndex and the data to be signed.

Output: The signatureValue, as specified in Clause 9, for the signature suite for the keyIndex's keyType.

7.2.6 DevID certificate enable/disable

Input: The certificateIndex of the DevID certificate and the desired state (enabled or disabled).

Output: None.

This operation allows the device administrator to disable use of a DevID without disabling the associated key, which might be in use by a different DevID. Re-enabling a disabled DevID certificate makes the DevID available once more, provided the associated key has not been disabled. The certificateIndex of a disabled DevID certificate is only valid as input to this operation and the LDevID certificate delete operation (7.2.13).

7.2.7 DevID key enable/disable

Input: The keyIndex of the DevID key and the desired state (enabled or disabled).

Output: None.

This operation allows the device administrator to control the use of DevID keys, and to maintain a measure of privacy by limiting exposure of the device's cryptographic identity (6.5). Disabling a DevID certificate or key can have authentication or authorization consequences as the device will no longer be able to assert its identity using the DevID for that certificate or any DevID that uses the key.

A disabled DevID key is not deleted or modified while 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 (7.2.10).

7.2.8 LDevID key generate

Input: The keyType, identifying the signature suite that will use the key.

Output: The keyIndex.

This operation allows the device administrator to generate an additional LDevID key within the DevID module. Newly generated keys are disabled and need to be explicitly enabled before use (7.2.7).

7.2.9 LDevID key insert

Input: The private key (DevID secret) to be inserted in the DevID module.

Output: The keyIndex.

This operation allows the device administrator to insert an externally generated LDevID key into the DevID module. Newly inserted keys are disabled and explicitly enabled before use (7.2.7); see 7.2.8 (LDevID key generate). The signature suite that will use the inserted private key is identified by that key.

7.2.10 LDevID key delete

Input: The keyIndex of the LDevID key to be deleted.

Output: None.

The DevID module may perform cryptographic zeroization on LDevID key storage as part of the delete process, removing both private and public key material.

After key deletion any LDevID certificates that make use of the key are 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 them.

This operation does not delete an IDevID key even if identified by the keyIndex.

7.2.11 LDevID certificate insert

Input: The certificate.

Output: The certificateIndex for the inserted certificate.

LDevID certificates can be installed at any time. Multiple LDevID certificates can be associated with the same DevID secret.

A DevID solution can use an IDevID presented by a device to generate a LDevID certificate, and can insert that certificate without requiring prior generation of a CSR by the device. The LDevID certificate can use the IDevID public key, allowing low-end devices to use LDevIDs without also supporting key generation.

The DevID module shall check that the certificate contains a public key that matches an existing DevID Module private key. The DevID module should check that the LDevID certificate meets the requirements specified in Clause 8. If either of these checks fails, the certificate is not inserted. Newly inserted LDevID certificates are disabled and explicitly enabled before use; see 7.2.6.

7.2.12 LDevID certificate chain insert

Input: The certificateIndex of an LDevID certificate and the certificate chain to be associated with that certificate.

Output: The certificateIndex for the inserted certificate.

7.2.13 LDevID certificate delete

Input: The certificateIndex of the certificate to be deleted.

Output: None.

This operation implicitly deletes any certificate chain associated with the deleted certificate, it does not remove the associated DevID secret. This operation does not delete an IDevID certificate even if identified by the certificateIndex.

The DevID module may perform cryptographic zeroization on LDevID certificate material as part of the delete process.

7.2.14 LDevID certificate chain delete

Input: The certificateIndex of the certificate associated with the certificate chain.

Output: None.

The DevID module may perform cryptographic zeroization on LDevID certificate chain material as part of the delete process.

7.2.15 Addition of RNG entropy

Input: 256 or fewer data octets.

Output: None.

This operation adds additional entropy to the RNG state to support the desired security strength.

7.3 DevID Management Interface

A local management agent can use the DevID management interface to support local or remote management operations, accessing all service interface operations (7.2) except initialization (7.2.1) and signing (7.2.5). The DevID management interface also allows access to operational statistics. An SMIv2 MIB that provides information on DevID module capabilities, statistics, and access to a subset of DevID operations, is defined in Clause 10.

8. DevID certificate fields and extensions

The certificates specified by this standard are a subset of the X.509 v3 certificates profiled in RFC 5280. This clause describes the fields used in DevID certificates and DevID intermediate certificates,[11] as summarized in Table 8-1 and Table 8-2. DevID solutions are expected to validate these certificates as specified in RFC 5280.

NOTE—A claim of conformance to this standard attests to the functionality of a device and information provided by a supplier of that device, not to all possible subsequent uses or configurations consistent with the specified functionality. Conformance (Clause 5) only references this clause to constrain the format of IDevID certificates and their certificate chains. However network administrators are warned that the use of LDevID certificates and certificate chains that do not meet the requirements of this clause can limit the functionality and use of DevID solutions.

Table 8-1—DevID certificate and intermediate certificate fields

Field name	RFC 5280 type	Value	Reference
`version`	`INTEGER`	3	8.1
`serialNumber`	`INTEGER`	Positive integer	8.2
`signature`	`AlgorithmIdentifier`	See Clause 9	8.3
`issuer`	`Name`	Name of issuing CA	8.4
`validity`	`UTCTime` or `GeneralizedTime`	`notBefore` (earliest) and `notAfter` (latest) use	8.5
`subject`	`Name`	Name of the DevID device	8.6
`subjectPublicKeyInfo`	`SubjectPublicKeyInfo`	The DevID public key	8.7
`signatureAlgorithm`	`AlgorithmIdentifier`	See Clause 9	8.8
`signatureValue`	See Clause 9	See Clause 9	8.9
`extensions`	`Extensions`	See Table 8-2	8.10

Table 8-2—DevID certificate and intermediate certificate extensions

Extension name	RFC 5280 type	Value	Reference
`authorityKeyidentifier`	`KeyIdentifier`	A unique value that matches the `subjectKeyIdentifier` of the issuer's certificate	8.10.1
`subjectKeyIdentifier` (not recommended in DevID certificates)	`KeyIdentifier`	See `authorityKeyIdentifier` above	8.10.2
`keyUsage`	`BIT STRING`	See RFC 5280	8.10.3
`subjectAltName`	`GeneralNames`	Additional name elements for the device	8.10.4

This clause requires that values of specified certificate fields be 'unique'. Unless otherwise specified, uniqueness is to be guaranteed within the certificate issuer's domain of significance.

[11]In this clause an unqualified reference to 'certificate' refers to both DevID certificates and DevID intermediate certificates (i.e., certificates in a DevID certificate chain), but not to X.509 certificates in general.

8.1 version

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

NOTE—In this clause, this `font` identifies ASN.1 encoded data including DER encoded certificate fields.

8.2 serialNumber

The `serialNumber` is a positive integer of up to 20 octets that identifies the certificate, is created by the CA that signs the certificate, and is unique in the scope of DevID certificates signed by that CA (see 6.4). A DevID certificate's `serialNumber` will typically be unrelated to any manufacturer serial numbers or other unique identifiers associated with the device.

8.3 signature

The identifier of the signature algorithm used to sign the certificate. The value of this field is identical to that of the `signatureAlgorithm` field (8.8).

8.4 issuer

The name used by the CA that has signed the certificate. This can be the name of the device supplier, manufacturer, a third party CA, a network operator, a standards certification agency, or an end user organization. It is expected to be 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 `issuer` field is most useful as a user interface mechanism and all certificates contain the `authorityKeyIdentifier` (8.10.1), and DevID intermediate certificates also contain the `subjectKeyIdentifier` (8.10.2). These extensions facilitate certificate path building, which is necessary to validate DevID certificates.

8.5 validity

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

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

The time the DevID is created is encoded in the `notBefore` field of DevID certificates. Each DevID intermediate certificate has a `notBefore` value that encodes a time that is the same as or prior to that of any DevID certificate that relies on the chain for certificate validation.

The latest time a DevID is expected to be used is encoded in the `notAfter` field of the DevID certificate. Each DevID intermediate certificate has a `notAfter` value that encodes a time that is the same as or later than that of any DevID certificate that relies on the chain for certificate validation.

Devices possessing an IDevID are expected to operate indefinitely into the future and should use the GeneralizedTime value 99991231235959Z[12] in the `notAfter` field of IDevID certificates. Solutions verifying a DevID are expected to accept this value indefinitely. Values in `notAfter` fields are treated as specified in RFC 5280.

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

8.6 subject

A certificate's `subject` and `subjectAltName` fields comprise a unique subject name that identifies the entity (a device in DevID certificates, and a CA in DevID intermediate certificates) associated with the public key in the `subjectPublicKeyInfo` field.

The DevID certificate `subject` field is always present, but can be empty. If non-null it contains an X.500 Distinguished Name (DN). An IDevID certificate `subject` field shall be non-null and should include a unique device serial number encoded as the `serialNumber` attribute (RFC 5280 `X520SerialNumber`). In the case of a third-party CA or a standards certification agency, the `subject` field can contain information identifying the supplier or manufacturer of the device.

NOTE 1—As stated above, the recommended `serialNumber` is unique within the issuer's (8.4) domain of significance, not just within the context of precursor DN fields.

NOTE 2—When a TPM is used to provide DevID module functionality, the IDevID certificate (created by the original equipment manufacturer) `subject` field contains a non-null X.500 DN identifying the device and the `subjectAltName` (8.10.4) identifies the TPM ([B13] 3.2).

RFC 5280 further constrains the use and encoding of the `subject` and `subjectAltName` fields in certificates whose subject is a CA (including DevID intermediate certificates).

8.7 subjectPublicKeyInfo

The `subjectPublicKeyInfo` field in DevID certificates and DevID intermediate certificates contains public key information for the Signature Suite that supports the DevID, as specified in Clause 9.

8.8 signatureAlgorithm

The `signatureAlgorithm` field in DevID certificates and DevID intermediate certificates identifies the CA signature algorithm used to sign this certificate as specified for the DevID's Signature Suite in Clause 9.

8.9 signatureValue

The signature of DevID certificates and DevID intermediate certificates is computed and encoded in the `signatureValue` field as specified for the DevID's Signature Suite in Clause 9.

8.10 extensions

The optional extensions field defined in RFC 5280 shall not be used for critical extensions in any IDevID certificate or IDevID intermediate certificate with the exception of the `keyUsage` extension (8.10.3).

This standard requires use of the `authorityKeyIdentifier` (8.10.1) non-critical extension in DevID and DevID intermediate certificates, and use of the `subjectKeyIdentifier` (8.10.1) non-critical extension in DevID intermediate certificates. It also recommends use of `subjectAltName` (8.10.1) in DevID certificates.

Use of the optional extensions field defined in RFC 5280 for critical extensions in LDevID certificates or intermediate certificates is not recommended. As specified in RFC 5280 a DevID solution that encounters a critical extension it does not understand will signal an error and fail validation of the LDevID.

8.10.1 authorityKeyIdentifier

To optimize building correct certificate chains, the non-critical Authority Key Identifier extension is populated with a unique value as recommended by RFC 5280 and is included in all certificates.

8.10.2 subjectKeyIdentifier

The Subject Key Identifier extension is included in each DevID intermediate certificate. This facilitates certificate chain building which is necessary to validate DevID certificates.

The Subject Key Identifier extension should not be included in DevID certificates. Its omission conserves space and certificate chain building does not make use of it, even if present.

8.10.3 keyUsage

The `keyUsage` field defined in RFC 5280 may be used 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 can 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`.

8.10.4 subjectAltName

The `subjectAltName` extension can be present in both DevID certificates and DevID intermediate certificates supplementing the subject name information in the `subject` (8.6) field as specified in RFC 5280.

If a DevID certificate includes a `subjectAltName`, that field should include a `HardwareModuleName` (as specified in RFC 4108) that provides additional information about the device.

NOTE—When a TPM is used to provide DevID module functionality the IDevID certificate (created by the original equipment manufacturer) contains a `subjectAltName` that uses a `HardwareModuleName` to identify the TPM ([B13] 3.2), with the `hwType` identifying the TPM Version (2.23.133.1.0 for Version 1.2) and the `hwSerialNum` containing the TPM Serial Number.

9. DevID signature suites

A signature suite is an interoperable specification of cryptographic algorithms used for signing (7.2.5) together with the values of parameters (for example, key types and sizes[13]) used by those algorithms, and their representation in DER encoded certificate fields.[14] Specification of the cryptographic signature functions that can be supported by a DevID module in terms of signature suites is intended to increase interoperability by providing a clear but limited number of alternatives. If a signature suite uses a combination of algorithms, their parameters are chosen so that the security provide by each complements the other(s)—the use of ECDSA P-256 with SHA-256 being an example.

The specific signature suites (Clause 9) that a DevID module uses to support its operations depend on the environment and existing security framework within which the device is to be used, on the characteristics of the device itself (available power and storage, for example), and on the security requirements (equivalent bits of security represented by the secret key(s) used, for example). A device can support the use of more than one signature suite to facilitate deployment in different environments.

This clause specifies each of the following DevID conformant signature suites:

— RSA-2048/SHA-256 (9.1)
— ECDSA P-256/SHA-256 (9.2)
— ECDSA P-384/SHA-384 (9.3)

by reference to other standards, together with the following:

a) The minimum security strength of keys (as specified in NIST SP 800-90A Revision 1) and any other requirements placed on a random number generator (RNG, 7.1.3) used to generate those keys for the signature suite within the DevID module (7.2.8).
b) Values of the `signatureAlgorithm` (8.8), `subjectPublicKeyInfo` (8.7), and `signatureValue` (8.9) fields of DevID and DevID intermediate certificates for that signature suite. In each case the `signature` (8.3) field takes the same value as the `signatureAlgorithm` field, as specified in 8.3.

The general structure of an `AlgorithmIdentifier` used in the `signatureAlgorithm` (8.8) field is defined in RFC 5280 4.1.1.2 by the ASN.1 sequence:

```
AlgorithmIdentifier ::= SEQUENCE {
       algorithm    OBJECT IDENTIFIER,
       parameters   ANY DEFINED BY algorithm OPTIONAL }
```

This `AlgorithmIdentifier` definition is also part of the general structure of the `subjectPublicKeyInfo` field defined in RFC 5280 4.1 by the sequence:

```
SubjectPublicKeyInfo ::= SEQUENCE {
       algorithm    AlgorithmIdentifier,
       subjectPublicKey    BIT STRING }
```

NOTE 1—The choice and combination of cryptographic methods is notorious for the introduction of unexpected security exposures. Each signature suite uses an algorithm or combination of algorithms whose interactions have been studied by the professional security community.

NOTE 2—A DevID module that supports more than one signature suite necessarily implements more than one IDevID. Conceptually these are merely different ways of expressing the same identity, though their security properties differ.

[13]And hence both the strength (resistance to attack) and the difficulty of implementation given a target time to complete operations.
[14]The underlying ASN.1 structures fully support algorithm agility.

9.1 RSA-2048/SHA-256

9.1.1 Algorithms and parameters

RSASSA-PKCS1-v1.5 signature schemes are defined in RFC 8017. The sha256WithRSAEncryption (RFC 4055) algorithm is used with a 2048-bit key and the SHA-256 secure hash algorithm as specified in NIST FIPS 180-4.

9.1.2 Key generation

An RNG used by a DevID module to generate keys for this signature suite shall have sufficient entropy to generate keys with a security strength of at least 128 bits.

9.1.3 signatureAlgorithm

The `signatureAlgorithm` field (8.8) value conforms to the general ASN.1 structure specified by RFC 5280 4.1.1.2 with the `algorithm` object identifier `sha256WithRSAEncryption` specified in RFC 4055 Section 5 and RFC 8017 A.2.4:

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

sha256WithRSAEncryption OBJECT IDENTIFIER ::=  { pkcs-1 11}
```

and a `parameters` field of type `NULL`, as specified in RFC 4055 Section 6 and RFC 8017 A.2.4.

9.1.4 subjectPublicKeyInfo

The `subjectPublicKeyInfo` field (8.7) conforms to the general ASN.1 structure specified by RFC 5280 4.1 with the `algorithm` object identifier `rsaEncryption`:

```
rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1}
```

and a `parameters` field of type `NULL`, as specified in RFC 3279 2.3.1 and RFC 8017 A.2.4.

The `subjectPublicKey BIT STRING` encapsulates the DER encoded `RSAPublicKey`, as specified in RFC 3279 2.3.1:

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

9.1.5 signatureValue

The `signatureValue` field (8.9) encodes the result of applying the signing algorithm as a `BIT STRING`.

9.2 ECDSA P-256/SHA-256

9.2.1 Algorithms and parameters

The ECDSA signature algorithm is defined in NIST FIPS 186-4 by reference to ANSI X9.62-2005. The SHA-256 message digest algorithm and the P-256 elliptic curve defined in FIPS 186-4 Annex D, D.1.2.3 are used.

9.2.2 Key generation

An RNG used by a DevID module to generate keys for this signature suite shall have sufficient entropy to generate keys with a security strength of at least 128 bits.

9.2.3 signatureAlgorithm

The `signatureAlgorithm` field (8.8) value conforms to the general ASN.1 structure specified by RFC 5280 4.1.1.2 with the `algorithm` object identifier `ecdsa-with-SHA256` as specified in RFC 5480 Appendix A:

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

The `parameters` field is omitted as specified in RFC 5480.

9.2.4 subjectPublicKeyInfo

The `subjectPublicKeyInfo` field (8.7) conforms to the general ASN.1 structure specified by RFC 5280 4.1 with the `algorithm` object identifier `id-ecPublicKey` and a `parameters` field with the object identifier `secp256r1` as specified in RFC 5480 2.1.1:

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


secp256r1 OBJECT IDENTIFIER  ::= { iso(1) member-body(2) us(840)
      ansi-x9-62(10045) curves(3) prime(1) 7 }
```

NOTE—`secp256r1` has the same value as `ansip256r1` specified in IEEE Std 802.1AR-2009.

The `subjectPublicKey BIT STRING` is a non-trivial encoding of `ECPoint`, as specified in RFC 5480 2.2:

```
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.

9.2.5 signatureValue

The ASN.1 structure for ECDSA signatures is specified in RFC 5480 and is included here for convenience:

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

9.3 ECDSA P-384/SHA-384

9.3.1 Algorithms and parameters

The ECDSA signature algorithm is defined in NIST FIPS 186-4 by reference to ANSI X9.62-2005. The SHA-384 message digest algorithm and the P-384 elliptic curve defined in FIPS 186-4 Annex D, D.1.2.3 are used.

9.3.2 Key generation

An RNG used by a DevID module to generate keys for this cipher suite shall have sufficient entropy to generate keys with a security strength of at least 192 bits.

9.3.3 signatureAlgorithm

The `signatureAlgorithm` field (8.8) value conforms to the general ASN.1 structure specified by RFC 5280 4.1.1.2 with the `algorithm` object identifier `ecdsa-with-SHA384` as specified in RFC 5480 Appendix A:

```
ecdsa-with-SHA384 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-sha2(3) 3 }
```

9.3.4 subjectPublicKeyInfo

The `subjectPublicKeyInfo` field (8.7) conforms to the general ASN.1 structure specified by RFC 5280 4.1 with the `algorithm` object identifier `id-ecPublicKey` and a `parameters` field with the object identifier `secp384r1` as specified in RFC 5480 2.1.1:

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


secp384r1 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
       certicom(132) curve(0) 34}
```

The `subjectPublicKey BIT STRING` is a non-trivial encoding of `ECPoint`, as specified in RFC 5480 2.2:

```
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.

9.3.5 signatureValue

The ASN.1 structure for ECDSA signatures is specified in RFC 5480 and is included here for convenience:

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

10. DevID MIB

This clause contains an SMIv2 Management Information Base (MIB) module, with read-only managed objects that describe DevID module capabilities and support inventory control and auditing operations (Clause 7, 7.3). This standard does not specify how DevIDs can be used to secure SNMP.

10.1 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 Simple Network Management Protocol (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 IETF STD 58 (RFC 2578), RFC 2579, and RFC 2580.

10.2 Relationship to other MIB modules

10.2.1 Relationship to the Entity MIB

An SNMP agent can manage a network element comprising one or many devices. They can include component devices (e.g., individual line cards in a chassis) or aggregate devices (such as the chassis and its current complement of cards). Some or all of these devices can incorporate a DevID module that binds DevIDs (secrets, certificates, and intermediate certificates) to the device whose identity can be asserted: they remain bound to a component device if it is removed from the network element, and are not retained by the SNMP agent.

A system implementing this MIB module shall implement the entPhysicalTable defined in RFC 6933, the Entity MIB. The entPhysicalIndex defined by the ENTITY-MIB identifies each physical device managed by the agent, and is used to index table entries with managed objects for each device with a DevID module, so ENTITY-MIB objects are correlated with and can supplement DevID module information cryptographically bound to devices. Entries in the entPhysicalTable include entPhysicalClass and entPhysicalContainedIn objects. These objects provide information on the general nature and containment relationships between multiple and aggregate devices within the scope of a single SNMP agent.

10.3 Structure of the MIB module

A single MIB module is defined in this clause. MIB objects are arranged in a number of tables, each table comprising SEQUENCE OF entries. Each table entry comprises a SEQUENCE of objects providing DevID information, and is identified by one or more index values, as follows:

- Each devIDModuleEntry in the devIDModuleTable table is identified by an entPhysicalIndex (10.2.1) value and describes the capabilities of a DevID module managed by an SNMP agent.
- Each devIDCertEntry in the devIDCertTable provides information for a DevID Certificate, and is identified by an entPhysicalIndex value (identifying the physical device whose DevID module contains that certificate) and a fingerprint of the certificate (devIDCertFingerprint).
- Each devIDChainEntry in the devIDChainTable provides information for a DevID intermediate certificate, with its index vales (entPhysicalIndex, devIDCertFingerprint, and devIDChainCertIndex) identifying its DevID module, the DevID certificate with which the certificate chain is associated, and the position on the chain (numbering upwards from the

DevIDcertificate, towards the trust anchor, with an index value of 1 for the first intermediate certificate).

— Each devIDStatisticsEntry in the devIDStatisticsTable provides audit information (7.3): counts of the number of times key pairs have been generated by, inserted into, or deleted from, the DevID module for the device identified by the entPhysicalIndex value, and counts of DevID certificate insertions and deletions.

DevID certificates and intermediate certificates are identified by a fingerprint, a Named Information Identifier (RFC 6920) that comprises a (possibly truncated) hash of the certificate preceded by a single octet that identifies the fingerprinting function and is one of the values recorded in the Named Information Hash Algorithm Registry. A certificate's fingerprint serves as a persistent identifier, independent of any local identifier such as the certificateIndex used by a DevID module's service interface (7.2) and any reorganization of that module's resources such as might occur when a certificate is deleted and reinserted or when one certificate is substituted for another.

NOTE—DevID module operations are not controlled by the local SNMP agent, and are not even necessarily visible through SNMP except as persistent changes to the information reported by the MIB module tables.

A certificate's fingerprint can be used to distinguish one certificate from another, or to identify associated intermediate certificates, without knowing the fingerprint function or being able to calculate the fingerprint. The DevID module can choose a fingerprint function that is conveniently related to a supported signature suite. A device administrator can also calculate a specific fingerprint in order to confirm use of an expected certificate.Similarly each DevID secret can be identified by a fingerprint of its `subjectPublicKeyInfo`.

Each devIDCertEntry and devIDChain entry includes the full certificate, allowing diagnosis of authentication or authorization failures due to the unexpected presence or absence of fields or field values (e.g., critical extensions). The `subjectPublicKeyInfo` field of a DevID module's IDevID certificates enumerates the signature suites supported by that module in addition to identifying specific keys.

10.4 Security considerations

SNMP versions prior to SNMPv3 did not include adequate security. Implementations that conform to this standard do not provide access to the functionality provided by this MIB using any version of SNMP prior to version 3 [5.3 f)] unless supported by TLS as specified by RFC 6353. Use of SNMP without invocation of cryptographic mechanisms (for authentication and privacy) is not consistent with the security goals of this standard.

This MIB does not provide write access to any MIB objects, or support their creation. All the MIB objects that can be read can be considered privacy sensitive. Any DevID certificate, certificate field, hash of the certificate, or hash of an associated public key could be used to track a personal device. The counts of DevID service operations maintained for auditing can be expected to change slowly and could contribute to an attempt to fingerprint the device. An IDevID's certificate and the DevID module's capabilities could provide information on the type of the device, and on when and where it was obtained. LDevID certificates intended for use in particular context might provide information on the device's use of particular networks and services, and a device owner might require additional controls beyond those specified for SNMP in general to restrict the exposure of certain DevID information. This MIB does not include any context specific access controls to individual DevID components.

Table 10-1—DevID managed objects*

Table	Table Entry objects	Object definitions and references
devIDModuleTable devIDModuleEntry [entPhysicalIndex] devIDMIBModuleGroup	 devIDModuleSupportsLDevIDs devIDModuleGeneratesLDevIDKeys devIDModuleInsertsLDevIDKeys	The DevID module for the device identified by entPhysicalIndex supports: LDevID operations 7.2 k)–n). LDevID key generation 7.2 h), j), 7.2.8,7.2.10. LDevID key insertion 7.2 i), j), 7.2.9, 7.2.10.
devIDCertTable devIDCertEntry [entPhysicalIndex, devIDCertFingerprint] devIDMIBCertificateGroup	devIDCertFingerprint devIDCertPublicKeyInfoFprint devIDCertIDevID devIDCertKeyEnabled devIDCertEnabled devIDCert	Identifies the DevID certificate 10.3. Identifies the DevID secret/keypair 10.3. True for an IDevID certificate 6.2, 7.3. Use of the Certificate's secret is enabled 7.2.7. Use of the Certificate's is enabled 7.2.6. The DevID certificate 6.2, 7.3.
devIDChainTable devIDModuleEntry [entPhysicalIndex, devIDCertFingerprint, devIDChainCertIndex] devIDMIBCertificateGroup	devIDChainCertIndex devIDChainCertFingerprint devIDChainCert	Position in certificate chain 10.3. Identifies the DevID certificate 10.3. The DevID intermediate certificate 6.3, 7.3.
devIDStatisticsTable devIDModuleEntry [entPhysicalIndex] devIDMIBStatisticsGroup	 devIDStatisticKeyGenerationCount devIDStatisticKeyInsertionCount devIDStatisticKeyDeletionCount devIDStatisticCertInsertionCount devIDStatisticCertDeletionCount	Counts of the number of times the following operations have been performed by the DevID module identified by entPhysicalIndex: Keypair generation 7.2.8. Keypair insertion 7.2.9. Keypair deletion 7.2.10. DevID certificate insertion 7.2.11. DevID certificate deletion 7.2.13.

*Table 10-1 excludes tables, table entries, groups and objects whose status is deprecated or obsolete. Refer to the MIB module for their definition and description, include reasons for deprecation or obsolescence.

10.5 Definitions for Secure Device Identifier MIB

In the following MIB definition, if there is any discrepancy between the DESCRIPTION text and the corresponding definitions in Clause 7 or Clause 8, the definitions take precedence.

```
-- ****************************************************************
-- IEEE8021-DEVID-MIB
--
-- Managed object definitions for IEEE 802.1AR Secure Device Identity
-- ****************************************************************

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    "201807151904Z"
    ORGANIZATION    "IEEE 802.1 Working Group"
    CONTACT-INFO    "WG-URL: http://www.ieee802.org/1
                     WG-EMail: stds-802-1-L@ieee.org

                       Contact: IEEE 802.1 Working Group Chair
                       Postal:  C/O IEEE 802.1 Working Group
                                IEEE Standards Association
                                445 Hoes Lane
                                Piscataway
                                NJ 08854
                                USA
                       E-mail:  STDS-802-1-L@IEEE.ORG"
    DESCRIPTION
    "The MIB module for managing an IEEE 802.1AR DevID (Secure Device
     Identifier) Module. A DevID comprises: a DevID secret (a private
     key) stored confidentially by the DevID module and accessible only
     through operations provided by the module; a DevID certificate
     containing the corresponding public key and a subject name that
     identifies the device; and a (possibly null) certificate chain. Use
     of the DevID module signing operations allows the device to prove
     possession of the DevID secret, and thus assert its identity in
     authentication protocols. An initial IDevID provided by the
     device supplier can be used directly or can be used to provision
     one or more locally significant LDevIDs that reflect authorization
     decisions by the local network administrator with certificate fields
     that record those decisions.
     An SNMP agent can manage a network element comprising one or
     many devices. They can include component (such as individual line
     cards in a chassis) or aggregate devices (such as the chassis and
     its current complement of cards). In each case a DevID module binds
     DevIDs secrets and certificates to the device whose identity they can be
     used to assert: they remain attached to a component device if it is
     removed from the network  element, and are not retained by the SNMP
     agent. The entPhysicalIndex defined by the ENTITY-MIB identifies each
     device managed by the agent and is used to index tables of managed
     objects for each device with a DevID module, so ENTITY-MIB objects are
     correlated with and can supplement DevID information cryptographically
     bound to the device.
     The initial version of this ieee8021DevIDMIB used the object name
     prefix 'devID' rather than 'ieee8021DevI' as recommended by
     RFC 4181. The 'devID' prefix has been retained for backwards
     compatibility and internal consistency."
    REVISION        "201807151904Z"
    DESCRIPTION
    "Published as part of IEEE Std 802.1AR-2018"
    REVISION        "200906250000Z"
    DESCRIPTION
    "Published as part of IEEE Std 802.1AR-2009"

     ::= { 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 }
                              -- unused (historic)

devIDMIBObjects        OBJECT IDENTIFIER   ::= { ieee8021DevIDMIB 1 }

devIDMIBConformance    OBJECT IDENTIFIER   ::= { ieee8021DevIDMIB 2 }

--
-- Textual Conventions - current
--
DevIDFingerprint ::= TEXTUAL-CONVENTION
     DISPLAY-HINT "1x:1x"
     STATUS             current
     DESCRIPTION "A Named Information identifier (RFC 6920) comprising a
```

```
     single octet (an IANA (iana.org) Named Information Hash Algorithm
     Registry value) followed by the result of applying that identified
     (possibly truncated) hash function  to the arbitrary long octet
     string to be fingerprinted. The fingerprint size (including the
     initial identifier) is limited to 49 octets to meet the SNMP oid
     size constraints when used as an INDEX while allowing the use of
     sha3-384, but sha-256-32 or sha-256-64 (5 or 9 octets total) is
     recommended with checking of full, not fingerprint, values in
     sensitive applications. This TEXTUAL-CONVENTION allows a zero-length
     value where the fingerprint value is optional. MIB definitions or
     implementations may refuse to accept a zero-length value."
     SYNTAX              OCTET STRING (SIZE (0 .. 49))
--
-- DevID Management Objects
--
devIDGlobalMIBObjects  OBJECT IDENTIFIER  ::= { devIDMIBObjects 1 }
                            -- unused (historic)

devIDMgmtMIBObjects    OBJECT IDENTIFIER  ::= { devIDMIBObjects 2 }

devIDStatsMIBObjectsOBJECT IDENTIFIER   ::= { devIDMIBObjects 3 }
                            -- unused (historic)
--
-- devIDMgmtMIBObjects - tables with current objects
--
devIDModuleTableOBJECT-TYPE
     SYNTAX              SEQUENCE OF DevIDModuleEntry
     MAX-ACCESS                          not-accessible
     STATUS                                   current
     DESCRIPTION
     "A table of DevID module capabilities, which can differ for devices
     managed by the same SNMP agent."
    REFERENCE"IEEE 802.1AR 7.3, 10.2, 10.3"
     ::= { devIDMgmtMIBObjects 6 }

devIDModuleEntryOBJECT-TYPE
     SYNTAX              DevIDModuleEntry
    MAX-ACCESS      not-accessible
    STATUS          current
     DESCRIPTION"DevID module capabilities, indexed by the ENTITY MIB's
     entPhysicalIndex."
     INDEX { entPhysicalIndex }
     ::= { devIDModuleTable 1 }

DevIDModuleEntry ::= SEQUENCE {
          devIDModuleSupportsLDevIDs TruthValue,
          devIDModuleGeneratesLDevIDKeys TruthValue,
          devIDModuleInsertsLDevIDKeys TruthValue
}

devIDModuleSupportsLDevIDsOBJECT-TYPE
     SYNTAX                                   TruthValue
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"True if the module supports the mandatory operations
    for LDevIDs."
    REFERENCE"IEEE 802.1AR 7.2(k)-(n)."
     ::= { devIDModuleEntry 1}

devIDModuleGeneratesLDevIDKeysOBJECT-TYPE
     SYNTAX                                   TruthValue
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"True if the module supports LDevID key generation."
    REFERENCE"IEEE 802.1AR 7.2(h), 7.2(j), 7.2.8, 7.2.10."
     ::= { devIDModuleEntry 2}

devIDModuleInsertsLDevIDKeysOBJECT-TYPE
     SYNTAX                                   TruthValue
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"True if the module supports LDevID key insertion."
```

```
    REFERENCE"IEEE 802.1AR 7.2(i), 7.2(j), 7.2.9, 7.2.10, 7.3."
     ::= { devIDModuleEntry 3}
--
devIDCertTable OBJECT-TYPE
     SYNTAX              SEQUENCE OF DevIDCertEntry
     MAX-ACCESS                          not-accessible
     STATUS                                   current
     DESCRIPTION"A table of DevID certificates, indexed by
     entPhysicalIndex (identifying the DevID module to which the
     certificate belongs) and the certificate's fingerprint."
    REFERENCE"IEEE 802.1AR Clause 6, 6.2, 7.2.2, 7.2.3, 7.2.6, 7.2.7,
    7.2.11, 7.2.13, 7.3."
     ::= { devIDMgmtMIBObjects 7 }

devIDCertEntry OBJECT-TYPE
     SYNTAX              DevIDCertEntry
     MAX-ACCESS     not-accessible
     STATUS              current
     DESCRIPTION"DevID certificate objects, indexed by entPhysicalIndex
     and its devIDCertFingerprint."
     INDEX{ entPhysicalIndex, devIDCertFingerprint }
     ::= { devIDCertTable 1}

DevIDCertEntry ::= SEQUENCE {
        devIDCertFingerprint            DevIDFingerprint,
        devIDCertPublicKeyInfoFprintDevIDFingerprint,
        devIDCertIDevID        TruthValue,
        devIDCertKeyEnabled        TruthValue,
        devIDCertEnabled        TruthValue,
        devIDCert           OCTET STRING
}

devIDCertFingerprint     OBJECT-TYPE
    SYNTAX               DevIDFingerprint
    MAX-ACCESS           not-accessible
    STATUS               current
    DESCRIPTION"A fingerprint of the DevID certificate, identifying the
    fingerprinting hash."
    REFERENCE   "IEEE 802.1AR 10.3"
     ::= { devIDCertEntry 1}

devIDCertPublicKeyInfoFprint OBJECT-TYPE
    SYNTAX               DevIDFingerprint
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"A fingerprint of the DevID certificate's
    subjectPublicKeyInfo field, identifying the fingerprinting hash."
    REFERENCE   "IEEE 802.1AR 10.3"
     ::= { devIDCertEntry 2}

devIDCertIDevID                    OBJECT-TYPE
     SYNTAX                                  TruthValue
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"True if this is an IDevID Certificate."
    REFERENCE"IEEE 802.1AR Clause 6, 6.2, 7.3."
     ::= { devIDCertEntry 3}

devIDCertKeyEnabled           OBJECT-TYPE
     SYNTAX                                  TruthValue
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"True if use of the DevID Secret for this certificate is
    enabled, allowing its use."
    REFERENCE"IEEE 802.1AR 7.2.7, 7.3"
     ::= { devIDCertEntry 4}

devIDCertEnabled              OBJECT-TYPE
     SYNTAX                                  TruthValue
    MAX-ACCESS           read-only
    STATUS               current
    DESCRIPTION"True if the certificate can be used."
```

```
    REFERENCE"IEEE 802.1AR 7.2.6"
     ::= { devIDCertEntry 5}

devIDCert                               OBJECT-TYPE
    SYNTAX              OCTET STRING
    MAX-ACCESS          read-only
    STATUS              current
    DESCRIPTION"The X.509 DevID certificate."
    REFERENCE   "IEEE 802.1AR 6.2, 7.3, Clause 8"
     ::= { devIDCertEntry 6}
--
devIDChainTable                   OBJECT-TYPE
     SYNTAX                              SEQUENCE OF DevIDChainEntry
     MAX-ACCESS                     not-accessible
     STATUS                              current
     DESCRIPTION"A table of DevID intermediate certificates, indexed by
     entPhysicalIndex (identifying the DevID module),
     devIDCertFingerprint (identifying the DevID certificate), and
     devIDChainCertIndex (identifying the certificate's position in
     the certificate chain, upwards from the DevID certificate)."
    REFERENCE"IEEE 802.1AR 10.3, 6.3, 7.2.3."
     ::= { devIDMgmtMIBObjects 8 }

devIDChainEntry                   OBJECT-TYPE
     SYNTAX                              DevIDChainEntry
     MAX-ACCESS                     not-accessible
     STATUS                              current
     DESCRIPTION"DevID intermediate certificate objects, indexed by
     entPhysicalIndex, DevID certificate fingerprint, and the
     certificate's position in the certificate chain."
     INDEX{ entPhysicalIndex, devIDCertFingerprint,
                devIDChainCertIndex }
     ::= { devIDChainTable 1}

DevIDChainEntry ::= SEQUENCE {
     devIDChainCertIndex     Unsigned32,
     devIDChainCertFingerprint           DevIDFingerprint,
     devIDChainCert               OCTET STRING
}

devIDChainCertIndex          OBJECT-TYPE
    SYNTAX              Unsigned32
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION"The position of this intermediate certificate in the
    certificate chain."
    REFERENCE   "IEEE 802.1AR 10.3."
     ::= { devIDChainEntry 1}

devIDChainCertFingerprint OBJECT-TYPE
    SYNTAX              DevIDFingerprint
    MAX-ACCESS          read-only
    STATUS              current
    DESCRIPTION"A fingerprint of the intermediate certificate, identifying the
    fingerprinting hash."
    REFERENCE   "IEEE 802.1AR 10.3."
     ::= { devIDChainEntry 2}

devIDChainCert                    OBJECT-TYPE
    SYNTAX              OCTET STRING
    MAX-ACCESS          read-only
    STATUS              current
    DESCRIPTION"The X.509 intermediate certificate in a certificate chain."
    REFERENCE   "IEEE 802.1AR 6.3, 7.3, Clause 8."
     ::= { devIDChainEntry 3}
--
devIDStatisticsTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DevIDStatisticsEntry
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION"Counts of selected operations for each DevID module."
    REFERENCE"IEEE 802.1AR 7.3."
```

```
    ::= { devIDMgmtMIBObjects 5 }

devIDStatisticsEntry OBJECT-TYPE
    SYNTAX              DevIDStatisticsEntry
    MAX-ACCESS          not-accessible
    STATUS                   current
    DESCRIPTION"Counts of selected operations for a DevID module."
    INDEX { entPhysicalIndex }
    ::= { devIDStatisticsTable 1 }

DevIDStatisticsEntry ::= SEQUENCE {
        devIDStatisticKeyGenerationCount        Counter32,
        devIDStatisticKeyInsertionCount         Counter32,
        devIDStatisticKeyDeletionCount          Counter32,
        devIDStatisticCSRGenerationCount        Counter32,
        devIDStatisticCredentialInsertionCountCounter32,
        devIDStatisticCredentialDeletionCount   Counter32,
        devIDStatisticCertInsertionCountCounter32,
        devIDStatisticCertDeletionCountCounter32
}

devIDStatisticKeyGenerationCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION"The number of LDevID key pairs generated by the
    module. Discontinuities at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR 7.2.8, 7.3."
    ::= { devIDStatisticsEntry 1 }

devIDStatisticKeyInsertionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION"The number of LDevID key pairs inserted into the module.
    Discontinuities occur at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR 7.2.9, 7.3."
    ::= { devIDStatisticsEntry 2 }

devIDStatisticKeyDeletionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION "The number of LDevID key pairs deleted by the module.
     Discontinuities occur at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR 7.2.10, 7.3."
    ::= { devIDStatisticsEntry 3 }

devIDStatisticCSRGenerationCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          deprecated
    DESCRIPTION"The number of Certificate Signing Requests (CSR,
    RFC2986) generated by the module.  Discontinuities occur at system
    restart and counter rollover. Deprecated: the module does not
    necessarily have all the information to generate a meaningful CSR,
    and key and certificate insertion is not tied to prior CSR
    generation. If required the signing operation can generate a CSR
    though this is not required for LDevID insertion."
    REFERENCE"IEEE 802.1AR-2009 6.4, and 6.3.11"
    ::= { devIDStatisticsEntry 4 }

devIDStatisticCredentialInsertionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The number of LDevID credential insertions.
     Discontinuities occur at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR-2009 6.4, and 6.3.12."
    ::= { devIDStatisticsEntry 5 }

devIDStatisticCredentialDeletionCount OBJECT-TYPE
```

```
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The number of LDevID credential deletions.
     Discontinuities occur at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR-2009 6.4, and 6.3.14."
    ::= { devIDStatisticsEntry 6 }

devIDStatisticCertInsertionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION"The number of LDevID certificate insertions.
     Discontinuities occur at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR 7.2.11, 7.3."
    ::= { devIDStatisticsEntry 7 }

devIDStatisticCertDeletionCount OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION"This number of LDevID certificate deletions.
     Discontinuities occur at system restart and counter rollover."
    REFERENCE"IEEE 802.1AR 7.2.13."
    ::= { devIDStatisticsEntry 8 }
--
-- devIDMIBConformance - current
--
devIDMIBCompliances  OBJECT IDENTIFIER
    ::= { devIDMIBConformance 1 }

devIDMIBGroups  OBJECT IDENTIFIER
    ::= { devIDMIBConformance 2 }

devIDMIBModuleCompliance2 MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION "Module Compliance for DevID MIB-2018."
    MODULE          --this module
    MANDATORY-GROUPS {
                    devIDMIBModuleGroup,
                    devIDMIBCertificateGroup,
                    devIDMIBAuditGroup
                    }
    ::= { devIDMIBCompliances 2 }


devIDMIBModuleGroup OBJECT-GROUP
    OBJECTS         {
                            devIDModuleSupportsLDevIDs,
                            devIDModuleGeneratesLDevIDKeys,
                            devIDModuleInsertsLDevIDKeys
}
    STATUS          current
    DESCRIPTION "DevID MIB objects describing module capabilities."
    ::= { devIDMIBGroups 2 }

devIDMIBCertificateGroup OBJECT-GROUP
    OBJECTS         {
                    devIDCertPublicKeyInfoFprint,
                        devIDCertIDevID,
                        devIDCertKeyEnabled,
                        devIDCertEnabled,
                        devIDCert,
                        devIDChainCertFingerprint,
                        devIDChainCert
}
    STATUS          current
    DESCRIPTION "DevID MIB objects for DevID public key,
    certificate, and certificate chain inventory operations."
    ::= { devIDMIBGroups 3 }

devIDMIBAuditGroup OBJECT-GROUP
```

```
    OBJECTS         {
                        devIDStatisticKeyGenerationCount,
                        devIDStatisticKeyInsertionCount,
                        devIDStatisticKeyDeletionCount,
                        devIDStatisticCertInsertionCount,
                        devIDStatisticCertDeletionCount
}
    STATUS          current
    DESCRIPTION "DevID MIB objects supporting DevID operation auditing."
    ::= { devIDMIBGroups 4 }
--**********************************************************************
-- Textual Conventions - obsolete
--
DevIDErrorStatus ::= TEXTUAL-CONVENTION
    STATUS obsolete
    DESCRIPTION "The error state  of a DevID operation."
    SYNTAX          INTEGER  {
                        none(1),
                        internalError(2)
                    }

DevIDAlgorithmIdentifier::= TEXTUAL-CONVENTION
    STATUS          obsolete
    DESCRIPTION "The algorithm type for the public key."
    SYNTAX          INTEGER  {
                        rsaEncryption(1),
                        idecPublicKey(2)
                    }
--
-- devIDMgmtMIBObjects - obsolete tables and individual objects
--
devIDPublicKeyCount OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION
    "The total number of DevID public keys installed in the module.
    Obsolete: the number of currently installed keys is the number of
    DevIDCertEntry's with the module's entPhysicalIndex and distinct
    devIDCertPublicKeyInfoFprint values, the maximum number can be an
    implementation dependent function of the keys' signature suites and
    the storage occupied by certificates and certificate chains."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    ::= { devIDMgmtMIBObjects 1 }
--
devIDPublicKeyTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DevIDPublicKeyEntry
    MAX-ACCESS      not-accessible
    STATUS          obsolete
    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.
     Obsolete: the public keys that have been installed and may be
     used can be obtained from the subjectPublicKeyInfo field in each of
     the DevIDCertEntry's devIDCert object."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    ::= { devIDMgmtMIBObjects 2 }

devIDPublicKeyEntry OBJECT-TYPE
    SYNTAX          DevIDPublicKeyEntry
    MAX-ACCESS      not-accessible
    STATUS          obsolete
    DESCRIPTION
     "An entry containing DevID public key, the keys keyIndex, a value
     indicating if the key is enabled.
     Obsolete: the public keys that have been installed and may be
     used can be obtained from the subjectPublicKeyInfo field in each of
     the DevIDCertEntry's devIDCert object. The table entry indexing did
     not support multiple key pairs per entPhysicalIndex."
```

```
    INDEX           { entPhysicalIndex }
    ::= { devIDPublicKeyTable 1 }

DevIDPublicKeyEntry ::= SEQUENCE {
    devIDPublicKeyIndexUnsigned32,
    devIDPublicKeyEnabledTruthValue,
    devIDPublicKeyAlgorithmDevIDAlgorithmIdentifier,
    devIDPublicKeyPubkeySHA1Hash    SnmpAdminString,
    devIDPublicKeyErrStatusDevIDErrorStatus
}

devIDPublicKeyIndex OBJECT-TYPE
    SYNTAX          Unsigned32 (0..4294967295 )
    MAX-ACCESS      not-accessible
    STATUS          obsolete
    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
     retrieving a key will be displayed in devIDPublicKeyErrStatus.
     Obsolete: the potential indexes are close packed forcing index
     reuse not under the agents control so reading the index from the
     devIDCredentialTable and then using it with this object may not
     retrieve the intended key."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    ::= { devIDPublicKeyEntry 1 }

devIDPublicKeyEnabled OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          obsolete
    DESCRIPTION "The enable/disable state of this public key. This
    setting persists across restarts. Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    ::= { devIDPublicKeyEntry 2 }

devIDPublicKeyAlgorithm OBJECT-TYPE
    SYNTAX          DevIDAlgorithmIdentifier
    MAX-ACCESS      read-only
    STATUS          obsolete
    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.
     Obsolete along with table. The AlgorithmIdentifier is not
     necessarily a complete description of the signature suite
     (parameters in subjectPublicKeyInfo may also be required), full
     information is in the devIDCert in the devIDCertTable using X.509
     OIDs so avoiding generating new OIDs for this MIB and removing the
     need for future MIB updates as new signature suites are added."
    REFERENCE
        "IEEE 802.1AR-2009 6.4, 6.3.2 and 7.2.9"
    ::= { devIDPublicKeyEntry 3 }

devIDPublicKeyPubkeySHA1Hash OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The SHA1 Hash of this DevID public key.
    Obsolete with table. DevIDFingerprint used in new table objects
    to provided allow hash flexibility without MIB update."
    REFERENCE
        "IEEE 802.1AR-2009 6.4, 6.3.2 and 7.2.9"
    ::= { devIDPublicKeyEntry 4 }

devIDPublicKeyErrStatus OBJECT-TYPE
    SYNTAX          DevIDErrorStatus
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "Displays the status of an operation on the public key.
    The default value is none which means no error, indicating a
    successful operation. Obsolete: DevID module service interface
```

```
    operations are not visible in this MIB so this object provides no
    clue as to what has failed and does not specify whether it is
    cleared by subsequent successful operations. If keys or certs are
    unusable they should not be visible to SNMP or appear not enabled. In
    both cases this read-only MIB cannot diagnose or repair. SNMP
    operations already have their own error codes."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    DEFVAL          { none }
    ::= { devIDPublicKeyEntry 5 }
--
devIDCredentialCount OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "This gives the total number of DevID credentials
    installed in the DevID module. Obsolete: Object is not indexed by
    entPhysicalIndex so is not a per module count if the agent is
    managing multiple devices. Changes as component devices are added
    or removed are not meaningful without other information. Per module
    counts can be obtained by interrogating the devIDCertTable."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    ::= { devIDMgmtMIBObjects 3 }
--
devIDCredentialTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DevIDCredentialEntry
    MAX-ACCESS      not-accessible
    STATUS          obsolete
    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 public key.
    Obsolete: the ASN.1 encoding of a certificate is already defined
    elsewhere, there is no need to require a device to decode the
    certificate into a different ASN.1 structure, and picking particular
    field might omit problematic extensions in particular certificates."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.3"
    ::= { devIDMgmtMIBObjects 4 }

devIDCredentialEntry OBJECT-TYPE
    SYNTAX          DevIDCredentialEntry
    MAX-ACCESS      not-accessible
    STATUS          obsolete
    DESCRIPTION "An entry containing DevID Credential information.
    Obsolete: Table entries are not indexed by entPhysicalIndex."
    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          obsolete
    DESCRIPTION "All credentials are indexed internally with this
    object. The value of this object is in  [0..devIDCredentialCount].
    Operations on credentials will use the credentialIndex to address a
    specific credential. The IDevID credential shall only be at index 0.
    Additional operations on credentials use the credentialIndex to
    address a specific credential.
    Obsolete: The SNP agent does not control or monitor individual
    DevID service operations, an SNMP agent can manage a system that
```

```
    comprises multiple devices identified by the ENTITY-MIB and more
    than one of those devices can have a DevID module with an IDevID. "
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    ::= { devIDCredentialEntry 1 }

devIDCredentialEnabled OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          obsolete
    DESCRIPTION "The enable/disable state of this credential. This
    setting persists across restarts. Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 6.3.6"
    ::= { devIDCredentialEntry 2 }

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

devIDCredentialSerialNumber OBJECT-TYPE
    SYNTAX          SnmpAdminString (SIZE  (0..20))
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The serial number of the credential.
    Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 7.2.2"
    ::= { devIDCredentialEntry 4 }

devIDCredentialIssuer OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The issuer field of the credential.
    Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 7.2.4"
    ::= { devIDCredentialEntry 5 }

devIDCredentialSubject OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The subject field of the credential.
    Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 7.2.8"
    ::= { devIDCredentialEntry 6 }

devIDCredentialSubjectAltName OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The subjectaltname field of the credential.
    Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 7.2.8"
    ::= { devIDCredentialEntry 7 }

devIDCredentialEntityIndex    OBJECT-TYPE
    SYNTAX          PhysicalIndex
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "This refers to the entPhysicalIndex in entPhysicalTable
    to identify the associated physical entity.
    Obsolete with table."
     REFERENCE       "IEEE 802.1AR-2009 6.4"
    ::= { devIDCredentialEntry 8 }

devIDCredentialPubkeyIndex OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
```

```
    STATUS          obsolete
    DESCRIPTION "Has the appropriate devIDPublicKeyIndex value from
    devIDPublicKeyTable to identify the public key information.
    Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 7.2.9"
    ::= { devIDCredentialEntry 9 }

devIDCredentialErrStatus OBJECT-TYPE
    SYNTAX          DevIDErrorStatus
    MAX-ACCESS      read-only
    STATUS          obsolete
    DESCRIPTION "The displays the status of an operation on the
    credential. The default value is none which means no error,
    indicating a successful operation.
    Obsolete with table."
    REFERENCE       "IEEE 802.1AR-2009 6.4, and 6.3.2"
    DEFVAL          { none }
    ::= { devIDCredentialEntry 10 }
--
-- devIDMIBConformance - obsolete
--
devIDMIBModuleCompliance MODULE-COMPLIANCE
    STATUS          obsolete
    DESCRIPTION "Module Compliance for DevID MIB-2009."
    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          obsolete
    DESCRIPTION "A collection of objects providing public key
    manageability, credential manageability and stats."
    ::= { devIDMIBGroups 1 }

--*********************************************************************
END
```

Annex A

(normative)

PICS proforma[15]

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

[15]*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-2018

Identification of protocol specification	IEEE Std 802.1AR-2018, 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 802.1AR-2018.)	No [ ] Yes [ ]

Date of Statement	

A.5 Major capabilities and options

Item	Feature	Status	References	Support
Does the DevID module:				
RSA	Support the RSA-2048/SHA-256 signature suite	O.1	5.3, 9.1, A.10	Yes [ ]
ECDSA256	Support the ECDSA P-256/SHA-256 signature suite	O.1	5.3, 9.2, A.11	Yes [ ]
ECDSA384	Support the ECDSA P-384/SHA-384 signature suite	O.1	5.3, 9.3, A.12	Yes [ ]
IDEVID	Contain an Initial Device Identifier (IDevID) for each of the supported signature suites	M	5.3a), Clause 6	Yes [ ]
IDSECRET	Contain an IDevID secret for each IDevID	M	5.3a), 6.1	Yes [ ]
IDCERT	Contain an IDevID certificate for each IDevID	M	5.3a), 6.2	Yes [ ]
IDCHAIN	Contain an IDevID certificate chain for each IDevID	M	5.3a), 6.3	Yes [ ]
STOR	Store DevID secrets as specified in	M	5.3b), 7.1	Yes [ ]
SI	Support the mandatory service interface operations for each supported signature suite	M	5.3c), 7.1, A.6	Yes [ ]
ICERTF	Include all mandatory fields in IDevID certificates	M	5.3d), A.8 Clause 8	Yes [ ]
ICHAINF	Include all mandatory fields in IDevID intermediate certificates	M	5.3d), A.8 Clause 8	Yes [ ]
XSNMP	Restrict SNMP access as specified	M	5.3f)	Yes [ ]
RNG	Generate random numbers for key generation or use by other protocol entities	SI-9:M O	5.3, 7.1.3 5.4d), A.7	Yes [ ] No [ ]
RNGSS	Use an RNG with the required security strength	M	5.3g), 7.1.3,	Yes [ ]
XSIGN	Permit remote access to signing operations	X	5.3h)	No [ ]
XCERTF	Include any prohibited field in an IDevID certificate or intermediate certificate	X	5.3i), 8.10, Clause 8	No [ ]
LDEVID	Support the use of at least one LDevID certificate	O	5.4a), 7.2, A.6	Yes [ ] No [ ]
ZKEY	Zeroize cryptographically when deleting keys	O	5.4c), 7.2.10	Yes [ ] No [ ]
ZCERT	Zeroize cryptographically when deleting certificates	O	5.4c), 7.2.13	Yes [ ] No [ ]
ZCHAIN	Zeroize cryptographically when deleting cert chains	O	5.4c), 7.2.14	Yes [ ] No [ ]
MIB	Support access to the DevID MIB using SNMP	MGT:O	5.4e), 7.3, Clause 10	Yes [ ] No [ ]

A.6 DevID Service Interface

Item	Feature	Status	References	Support
Does the DevID module implement the following operations:				
SI-I	Initialization	M	7.2.1	Yes [ ]
SI-KN	DevID public key enumeration	M	7.2.2	Yes [ ]
SI-CN	DevID certificate enumeration	M	7.2.3	Yes [ ]
SI-CHN	DevID certificate chain enumeration	M	7.2.4	Yes [ ]
SI-S	Signing	M	7.2.5	Yes [ ]
SI-CE	DevID certificate enable/disable	M	7.2.6	Yes [ ]
SI-KE	DevID key enable/disable	M	7.2.7	Yes [ ]
SI-KG	LDevID key generate	LDEVID:O	7.2.8	Yes [ ] No [ ]
SI-KI	LDevID key insert	LDEVID:O	7.2.9	Yes [ ] No [ ]
SI-KD	LDevID key delete	SI-9:M SI-10:M	7.2.10	Yes [ ] N/A[ ]
SI-CI	LDevID certificate insert	LDEVID:M	7.2.11	Yes [ ] N/A[ ]
SI-CHI	LDevID certificate chain insert	LDEVID:M	7.2.12	Yes [ ] N/A[ ]
SI-CD	LDevID certificate delete	LDEVID:M	7.2.13	Yes [ ] N/A[ ]
SI-CCD	LDevID certificate chain delete	LDEVID:M	7.2.14	Yes [ ] N/A[ ]
SI-RNT	Addition of RNG entropy	RNG:O	7.2.15	Yes [ ] No [ ]

A.7 DevID Random number generation

Item	Feature	Status	References	Support
Does the DevID module:				
RNG-N	Use a non-deterministic RNG	RNG:O2	5.4d), 7.1.3	Yes [ ]
RNG-D	Use a deterministic RNG	RNG:O2	5.4d), 7.1.3	Yes [ ]
DRBG	Use an SP 800-90A Revision 1 DRBG for the RNG	RNG-D: O	5.4d) 1), 7.1.3	A_
RNTRP	Support addition of RNG entropy	RNG-D:O SI-RNT: M	5.4d) 2), 7.1.3	Yes [ ]

A.8 DevID Certificate fields and extensions

Item	Feature	Status	References	Support
Does each IDevID certificate in the DevID module:				
NAME	Have a subject name that has not been and will not be issued to any other device by the CA issuing the certificate	M	5.3a), 6.1	Yes [ ]
DNAME	Have an X.500 DN in the `subject` field.	M	5.3e), 8.6	Yes [ ] No [ ]
SNAME	Include a unique device `serialNumber` in the `subject` field of each IDevID certificate.	O	5.4b), 8.6	Yes [ ] No [ ]
ALTNAME	Include the `subjectAltName` in each IDevID certificate.	O	5.4f), 8.10.4	Yes [ ] No [ ]
HWNAME	Include the device `HardwareModuleName` in the `subjectAltName` of each IDevID certificate.	O	5.4g), 8.10.4	Yes [ ] No [ ]
CERTF1	Contain the `subjectKeyIdentifier`	O	8.10.2	No [ ] Yes [ ]
Does each IDevID certificate and intermediate certificate in the DevID module:				
CERTF2	Contain the `version`, `serialNumber`, `signature`, `issuer`, `validity`, `subjectPublicKeyInfo`, `signatureAlgorithm`, and `signatureValue`, and `authorityKeyIdentifier` as specified in Clause 8.	M	5.3d), 8.1, 8.2,8.3, 8.4, 8.5, 8.7, 8.8, 8.9, 8.10.1	Yes [ ]
Does each intermediate certificate in the DevID module:				
CERTF3	Contain the `subjectKeyIdentifier`	M	8.10.2	Yes [ ]

A.9 DevID Supplier Information

Item	Feature	Status	References	Support
ESTOR	What encryption mechanism is used if external storage is supported?	O M	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 appropriate for authorization, auditing, or other purposes? How are these parsed?	M	Clause 6	S_

A.10 RSA-2048/SHA-256 Signature Suite

Item	Feature	Status	References	Support
S1-A	Does the DevID module support RSA-2048/SHA-256 with the algorithms and parameter types specified?	RSA:M	9.1.1	Yes [ ]
S1-KS	Do any keys generated by the DevID module for this signature suite have the specified strength?	RSA:M	9.1.2	Yes [ ]
S1-CSA	Are signatureAlgorithm values in DevID certificates and intermediate certificates as specified for this suite?	RSA:M	9.1.3	Yes [ ]
S1-CPK	Are signatureAlgorithm values in DevID certificates and intermediate certificates as specified for this suite?	RSA:M	9.1.4	Yes [ ]
S1-CSV	Are the signatureValue values in DevID certificates and intermediate certificates as specified for this suite?	RSA:M	9.1.5	Yes [ ]

A.11 ECDSA P-256/SHA-256 Signature Suite

Item	Feature	Status	References	Support
S2-A	Does the DevID module support ECDSA P-256/SHA-256 with the algorithms and parameter types specified?	ECDSA256:M	9.2.1	Yes [ ]
S2-KS	Do any keys generated by the DevID module for this signature suite have the specified strength?	ECDSA256:M	9.2.2	Yes [ ]
S2-CSA	Are signatureAlgorithm values in DevID certificates and intermediate certificates as specified for this suite?	ECDSA256:M	9.2.3	Yes [ ]
S2-CPK	Are signatureAlgorithm values in DevID certificates and intermediate certificates as specified for this suite?	ECDSA256:M	9.2.4	Yes [ ]
S2-CSV	Are the signatureValue values in DevID certificates and intermediate certificates as specified for this suite?	ECDSA256:M	9.2.5	Yes [ ]

A.12 ECDSA P-384/SHA-384 Signature Suite

Item	Feature	Status	References	Support
S3-A	Does the DevID module support ECDSA P-256/SHA-256 with the algorithms and parameter types specified?	ECDSA384:M	9.3.1	Yes [ ]
S3-KS	Do any keys generated by the DevID module for this signature suite have the specified strength?	ECDSA384:M	9.3.2	Yes [ ]
S3-CSA	Are signatureAlgorithm values in DevID certificates and intermediate certificates as specified for this suite?	ECDSA384:M	9.3.4	Yes [ ]
S3-CPK	Are signatureAlgorithm values in DevID certificates and intermediate certificates as specified for this suite?	ECDSA384:M	9.3.4	Yes [ ]
S3-CSV	Are the signatureValue values in DevID certificates and intermediate certificates as specified for this suite?	ECDSA384:M	9.3.5	Yes [ ]

Annex B

(informative)

Scenarios for DevID

DevIDs are used by cryptographic identification and authentication protocols. This standard describes the use of DevIDs with existing provisioning and authentication protocols. It does not provide a normative definition of such a protocol.

B.1 DevID use in EAP-TLS

Network devices are often identified using an IEEE 802.1X [B1] EAP method. EAP methods, such as EAP_TLS, can use X.509 client credentials including DevIDs.

A device that uses a DevID in an EAP-TLS exchange can be expected to use the operations defined in 7.2 as follows:

a) Certificates are enumerated (7.2.3) to determine the available and enabled DevID certificates. How the EAP-TLS implementation chooses the appropriate DevID certificate is specific to the device and its use of the authentication protocol.

b) Signing (7.2.3) is used 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 B-1 assuming that the authenticator also provides the authentication service. In the second request-response protocol exchange, 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, 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 protocol use. 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. Validation of the contents of the decrypted finished message establishes the authenticity of all of the messages the client sent and received (received as-sent, without error). 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 validated each other’s identity using the DevID certificate and DevID secret.

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

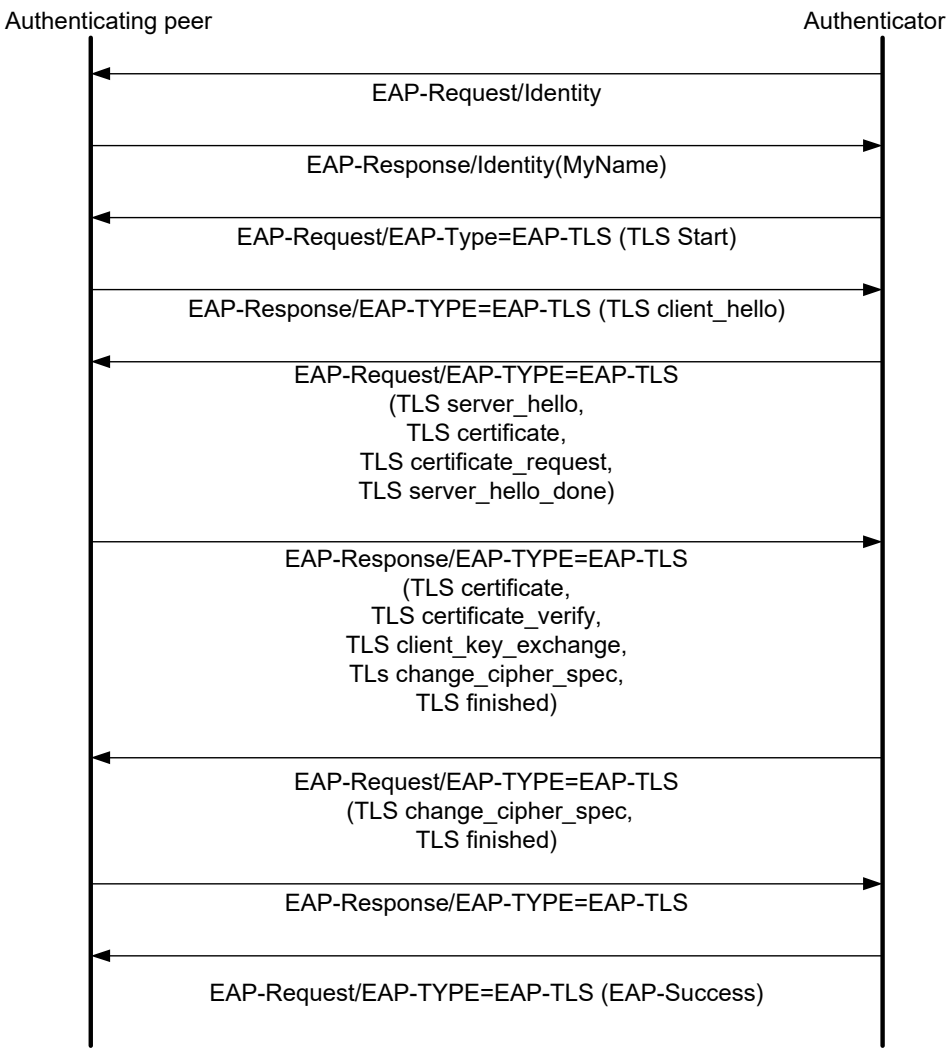

Figure B-1—Example EAP-TLS exchange

B.2 DevID uses in consumer devices

End users are not expected to use DevIDs directly. Instead the DevID is expected to be used by a home networking access point or router. These devices provide network connectivity to 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. A DevID can be used as a secure 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.

B.3 DevID uses in enterprise devices

In an enterprise environment an IDevID can 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 usage.

An 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 is supposed.

Annex C

(informative)

Bibliography

[B1] IEEE Std 802.1X™, IEEE Standard for Local and metropolitan area networks—Port-Based Network Access Control.[16, 17]

[B2] IEEE Std 1363™-2000, IEEE Standard Specifications for Public-Key Cryptography.

[B3] IETF RFC 3410, Introduction and Applicability Statements for Internet Standard Management Framework, Case, J., Mundy, R., Partain, D., Stewart, B., December 2002.

[B4] IETF RFC 3748, Extensible Authentication Protocol (EAP), L. Blunk, J. Vollbrecht, B. Aboba, J. Carlson, H. Levkowetz, June 2004.

[B5] IETF RFC 4086, Randomness Requirements for Security, Eastlake 3rd, D., Schiller, J., Crocker, S., June 2005.

[B6] IETF RFC 5216, The EAP-TLS Authentication protocol, D. Simon, B. Aboba, R. Hurst, March 2008.

[B7] IETF RFC 5246, The Transport Layer Security (TLS) Protocol, Version 1.2, T Dierks, E. Resoria, August 2008.

[B8] IETF RFC 5915, Elliptic Curve Private Key Structure.

[B9] IETF RFC 7030, Enrollment over Secure Transport, Pritikin, M., Yee, P., Harkins, D., October 2013.

[B10] IETF RFC 7170, Tunnel Extensible Authentication Protocol (TEAP Version 1, Zhou, H.,Cam-Winget, N., Salowey, J., Hanna, S., May 2014.

[B11] NIST FIPS 140-2, Security Requirements for Cryptographic Modules, 2002 December 3.[18]

[B12] NIST Special Publication 800-57, Recommendation for Key Management—Part 1: General (Revision 4), January 2016.

[B13] TCG Infrastructure WG, TPM Keys for Platform Identity for TPM 1.2 Specification Version 1.0, Revision 3, August 2015.[19]

[B14] Bootstrapping Remote Secure Key Infrastructures (BRSKI), Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., Watsen, K.[20]

[B15] Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices, Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman.[21]

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

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

[18]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).

[19]http://www.trustedcomputinggroup.org/wp-content/uploads/TPM_Keys_for_Platform_Identity_v1_0_r3_Final.pdf

[20]https://tools.ietf.org/html/draft-ietf-anima-bootstrapping-keyinfra

[21]https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final228.pdf

Consensus

WE BUILD IT.

Connect with us on:

Facebook: https://www.facebook.com/ieeesa

Twitter: @ieeesa

LinkedIn: http://www.linkedin.com/groups/IEEESA-Official-IEEE-Standards-Association-1791118

IEEE-SA Standards Insight blog: http://standardsinsight.com

YouTube: IEEE-SA Channel

IEEE
standards.ieee.org
Phone: +1 732 981 0060 Fax: +1 732 562 1571