IEEE SA
STANDARDS
ASSOCIATION

STANDARDS

IEEE Standard for
Local and Metropolitan Area Networks—

Port-Based Network Access Control

IEEE Computer Society

Developed by the
LAN/MAN Standards Committee

IEEE Std 802.1X™-2020
(Revision of IEEE Std 802.1X-2010
Incorporating IEEE Std 802.1Xbx™-2014
and IEEE Std 802.1Xck™-2018)

IEEE

IEEE Std 802.1X™-2020
(Revision of IEEE Std 802.1X™-2010
Incorporating IEEE Std 802.1Xbx™-2014
and IEEE Std 802.1Xck™-2018)

IEEE Standard for Local and Metropolitan Area Networks—

Port-Based Network Access Control

Developed by the

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 30 January 2020

IEEE SA Standards Board

Abstract: Port-based network access control allows a network administrator to restrict the use of IEEE 802® LAN service access points (ports) to secure communication between authenticated and authorized devices. This standard specifies a common architecture, functional elements, and protocols that support mutual authentication between the clients of ports attached to the same LAN and that secure communication between the ports, including the media access method independent protocols that are used to discover and establish the security associations used by IEEE 802.1AE™ MAC Security.

Keywords: access control, authentication, authorization, controlled port, EAP, EAPOL, IEEE 802.1X, key agreement, LANs, local area networks, MACsec, MACsec Key Agreeement, MAC security, MAC Service, MANs, metropolitan area networks, MKA, port-based network access control, secure association, security, service access point, uncontrolled port

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

PDF: ISBN 978-1-5044-6440-6 STD24052
Print: ISBN 978-1-5044-6441-3 STDPD24052

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 Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from the IEEE or viewed at https://standards.ieee.org/ipr/disclaimers.html.

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. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups 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 Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

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 SA Website at http://ieeexplore.ieee.org/browse/standards/collection/ieee 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 submitted to the IEEE SA Standards Board for approval, the IEEE 802.1 working group had the following membership:

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

Astrit Ademaj
Ralf Assmann
Jens Bierschenk
Christian Boiger
Paul Bottorff
Radhakrishna Canchi
Feng Chen
Weiying Cheng
Paul Congdon
Rodney Cummings
Josef Dorr
Hesham Elbakoury
Thomas Enzinger
János Farkas
Donald Fedyk
Norman Finn
Geoffrey Garner
Craig Gunther
Marina Gutierrez
Stephen Haddock
Mark Hantel
Marc Holness
Satoko Itaya
Yoshihiro Ito
Michael Karl
Stephan Kehrer
Randy Kelsey
Hajime Koto
James Lawlis
Christophe Mangin
Scott Mansfield
Kenichi Maruhashi
David McCall
Larry McMillan
Tero Mustala
Roy Myers
Hiroki Nakano
Bob Noseworthy
Tomoki Ohsawa
Hiroshi Ohue
Donald R. Pannell
Michael Potts
Dieter Proell
Wei Qiu
Karen Randall
Maximilian Riegel
Jessy V. Rouyer
Atsushi Sato
Frank Schewe
Maik Seewald
Johannes Specht
Marius Stanica
Guenter Steindl
Karim Traore
Hao Wang
Xinyuan Wang
Tongtong Wang
Ludwig Winkel
Karl Weber
Brian Weis
Jordon Woods
Nader Zein
Takahiro Yamaura
Helge Zinner
William Zhao
Harald Zweck

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
Harry Bims
Kenneth Bow
Rich Boyer
Nancy Bravin
Vern Brethour
Matthew Brown
Demetrio Bucaneg, Jr.
William Byrd
Radhakrishna Canchi
Paul Cardinal
Juan Carreon
Janos Farkas
Avraham Freedman
Devon Gayle
Tim Godfrey
Zhigang Gong
Randall Groves
Marek Hajduczenia
Marco Hernandez
Werner Hoelzl
Yasuhiro Hyakutake
Raj Jain
SangKwon Jeong
Pranav Jha
Lokesh Kabra
Srinivas Kandala
Piotr Karocki
Stuart Kerry
Evgeny Khorov
Yongbum Kim
Hyeong Ho Lee
Suzanne Leicht
James Lepp
Michael Lynch
John Mackay
Jouni Malinen
Roger Marks
Arthur Marris
Stephen McCann
Brett Mcclellan
Richard Mellitz
Michael Montemurro
Nick S.A. Nikjoo
Satoshi Obara
Robert O'Hara
Carlos Pardo
Bansi Patel
Arumugam Paventhan
David Piehler
Walter Pienciak
Clinton Powell
Karen Randall
R. K. Rannow
Maximilian Riegel
Robert Robinson
Jessy Rouyer
John Sargent
Frank Schewe
Michael Seaman
Thomas Starai
Walter Struppler
Michael Thompson
Mark-Rene Uchida
Alexander Umnov
Dmitri Varsanofiev
George Vlantis
Lisa Ward
Stephen Webb
Brian Weis
Scott Willy
Chun Yu Charles Wong
Oren Yuen

When the IEEE SA Standards Board approved this standard on 30 January 2020, it had the following membership:

Gary Hoffman, *Chair*
Vacant, *Vice-Chair*
Jean-Philippe Faure, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
Doug Edwards
Travis Griffith
Grace Gu
Guido Hiertz
John Kulick
David J. Law
Joseph L.Koepfinger*
Howard Li
Dong Liu
Kevin Lu
Paul Nikolich
Damir Novosel
Jon Rosdahl
Dorothy Stanley
Mehmet Ulema
Lei Wang
Sha Wei
Philip Winston
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1X™-2020, IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control.

This edition of IEEE Std 802.1X™ incorporates IEEE Std 802.1X-2010 and its amendments, IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018.

The first edition of IEEE Std 802.1X™ was published in 2001. The second edition, IEEE Std 802.1X-2004, clarified mutual authentication and the interface between the IEEE 802.1X state machines and the Extensible Authentication Protocol (EAP) and by IEEE Std 802.11™ in support of IEEE Std 802.1X.

The third edition, IEEE Std 802.1X-2010, added authenticated key agreement in support of IEEE Std 802.1AE™ MAC Security (MACsec) and clarified and generalized the relationship between the common architecture specified for port-based network access control and the functional elements and protocols that support that architecture as specified in IEEE Std 802.1X, other IEEE 802® standards, and IETF RFCs. Further changes updated the standard to reflect best current practice, insisting, for example, on mutual authentication methods and using such methods in examples. A greater emphasis was placed on the security of systems accessing the network, as well as on the security of the network accessed, with a more comprehensive treatment of segregating and limiting connectivity to unauthenticated systems. Applications of port-based network access that use MACsec and/or MACsec Key Agreement protocol (MKA) were described.

IEEE Std 802.1X-2010 included a number of improvements to the specification of the port access control protocol (PACP) state machines and their relationship to EAP methods and state machines. Systems conformant to IEEE Std 802.1X-2020 or IEEE Std 802.1X-2010 should interoperate, without prior configuration, with implementations conforming to IEEE Std 802.1X-2004 and IEEE Std 802.1X-2001. However, it is anticipated that claims of conformance with respect to some existing implementations, not needing to support IEEE Std 802.1AE and already conforming to best current practice as of 2010, will continue to refer to IEEE Std 802.1X-2004.

The first amendment to IEEE Std 802.1X-2010, IEEE Std 802.1Xbx-2014, extended MKA to further support and use the extended packet numbering Cipher Suites specified by the IEEE Std 802.1AEbw™-2013. Secure connectivity association (CA) members can temporarily suspend MKA operation without causing protocol timeouts that would disrupt secure data transfer, thus allowing in-service control plane software upgrades.

The second amendment to IEEE Std 802.1X-2010, IEEE Std 802.1Xck-2018, specified a YANG data model for configuration and status reporting, using the information model previously specified in this standard.

Contents

Figures

Tables

IEEE Standard for
Local and Metropolitan Area Networks—

Port-Based Network Access Control

1. Overview

1.1 Scope

For the purpose of providing compatible authentication, authorization, and cryptographic key agreement mechanisms to support secure communication between devices connected by **IEEE 802**® Local Area Networks (LANs), this standard

a) Specifies a general method for provision of port-based network access control.

b) Specifies protocols that establish secure associations for IEEE Std 802.1AE™ MAC Security.

c) Facilitates the use of industry standard authentication and authorization protocols.

1.2 Purpose

IEEE 802 LANs are deployed in networks that convey or provide access to critical data, that support mission critical applications, or that charge for service. Protocols that configure, manage, and regulate access to these networks and network-based services and applications typically run over the networks themselves. Port-based network access control regulates access to the network, guarding against transmission and reception by unidentified or unauthorized parties, and consequent network disruption, theft of service, or data loss.

1.3 Introduction

The stations attached to an IEEE 802 LAN transmit and receive data frames using the service provided by the IEEE 802 LAN MAC at a service access point, often referred to as a port, within each end station or bridge. Port-based network access control specifies a common architecture comprising cooperative functional elements and protocols that

a) Use the service provided by the LAN MAC, at a common service access point, to support a Controlled Port that provides secure access-controlled communication and an Uncontrolled Port that supports protocols that initiate the secure communication or do not require protection.

b) Support mutual authentication between a Port Access Entity (PAE) associated with a Controlled Port, and a peer PAE associated with a peer port in a LAN attached station that desires to communicate through the Controlled Port.

c) Secure the communication between the Controlled Port and the authenticated peer port, excluding other devices attached to or eavesdropping on the LAN.

d) Provide the Controlled Port with attributes that specify access controls appropriate to the authorization accorded to the peer station or its user.

This standard specifies the use of EAP, the Extensible Authentication Protocol (IETF RFC 3748 [B14][1]), to support authentication using a centrally administered Authentication Server and defines EAP encapsulation over LANs (EAPOL, Clause 11) to convey the necessary exchanges between peer PAEs attached to a LAN.

Where communication over the LAN connecting a Controlled Port to its peer(s) is physically secure, no additional protocol is required to protect their communication. This mode of operation is supported by this standard. More commonly intrusion into the LAN communication is a principal security threat, and the result of mutual authentication is not simply Controlled Port authorization to transmit and receive data, but secure distribution of master keys and associated data to the communicating peers. Proof of possession of master keys subsequently serves as proof of mutual authentication in key agreement protocols. These protocols generate keys that are used to cryptographically protect data frames transmitted and received by the Controlled Port. IEEE Std 802.11™ Wireless LANs specifies protocols that associate wireless stations with access points and initiate mutual authentication using the procedures specified in this standard, the subsequent generation of keys to protect data transfer, and the cryptographic methods that protect data frames using those keys. IEEE Std 802.1AE MAC Security (MACsec) specifies cryptographic support of the Controlled Port for other media access methods. Authenticated key agreement for MAC Security, as specified in this standard, specifies the generation of the Secure Association Keys (SAKs) used by MACsec.

Use of the Controlled Port can be restricted by access controls bound to the results of authentication and distributed via AAA protocols such as Diameter (IETF RFC 6733 [B25]) or RADIUS (IETF RFC 2865 [B6]). Attributes supporting certain port-based network access control scenarios are described in IETF RFC 3580 [B13], IETF RFC 4675 [B17], IETF RFC 4849 [B18], IETF RFC 7268 [B28], and IETF RFC 8044 [B29].

Clause 7 illustrates use of the above components and protocols in typical network access control scenarios.

1.4 Provisions of this standard

The scope (1.1) of this standard is addressed by detailed specification of the following:

a) The principles of port-based network access control operation, identifying the protocol components that compose a port-based network access control implementation (Clause 6).

b) A PAE component, that supports authentication, authorization, and the key agreement functionality required by IEEE Std 802.1AE to allow a MAC Security Entity (SecY) to protect communication through a port (6.3, Clause 12).

c) A Port Access Controller (PAC) component, that controls communication where the attached LAN is deemed to be physically secure and provides point-to-point connectivity (6.4).

d) The key hierarchy used by the PAE and SecY (6.2).

e) The use of EAP by PAEs to support authentication and authorization using a centrally administered Authentication or AAA Server (Clause 8).

f) An encapsulation format, EAPOL, that allows EAP Messages and other protocol exchanges to support authentication and key agreement to be carried directly by a LAN MAC service (Clause 11).

g) A MAC Security Key Agreement protocol (MKA) that the PAE uses to discover associations and agree the keys used by a SecY (Clause 9).

h) An EAPOL Announcement protocol that allows a PAE to indicate the availability of network services, helping other PAEs to choose appropriate credentials and parameters for authentication and network access (Clause 10).

i) Requirements for management of port-based access control, identifying the managed objects and defining the management operations for PAEs (12.9).

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

j) SMIv2 MIB objects that can be used with SNMPv3 to manage PAEs (Clause 13).

k) YANG configuration and operational state models for PAE and PAE System components (Clause 14).

The use of port-based network access control in a number of applications is described (Clause 7) to illustrate the use of these components and the requirements taken into account in their specification. To facilitate migration to this standard, Annex F (informative) uses the same concepts to describe the architectural modeling of unsecured multi-access LANs, a widely deployed form of authenticated port-based network access control that does not meet the security requirements of this standard. Administrative connectivity to unauthenticated devices, as required for use of industry standard 'Wake-on-LAN' (WoL) protocols, is described for the scenarios of Clause 7; Annex E (informative) provides background information on WoL.

This standard defines conformance requirements (Clause 5) for the implementation of the following:

l) Port Access Entities (PAEs)

m) Port Access Controllers (PACs)

Annex A provides PICS (Protocol Implementation Conformance Statement) Proformas for completion by suppliers of implementations that are claimed to conform to this standard.

The basic architectural concepts, such as 'port', on which this standard relies are reviewed in IEEE Std 802.1AC.

This standard uses and selects options provided by EAP and AAA protocol specifications, but does not modify those specifications (see Clause 2 for references). Annex D (informative) provides EAP and RADIUS usage guidelines.

The specification and conformance requirements for association discovery and key agreement for IEEE 802.11 Wireless LANs are outside the scope of this standard (see IEEE Std 802.11). That standard makes use of the PAE specified by this standard.

2. Normative references

The following referenced documents are indispensable for the application of this document (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.

DMTF, Alert Standard Format (ASF) Specification, Version 2.0, 23 April 2003.[2]

iana-if-type YANG Module, Internet Assigned Numbers Authority.[3]

IEEE Std 802®, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.[4, 5]

IEEE Std 802d™, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture Amendment 1: Allocation of Uniform Resource Name (URN) Values in IEEE 802 Standards.

IEEE Std 802.1Q™, IEEE Standard for Local and Metropolitan Area Networks: Bridges and Bridged Networks.

IEEE Std 802.1AB™, IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity and Discovery.

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

IEEE Std 802.1AE™, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security.

IEEE Std 802.1AX™, IEEE Standard for Local and Metropolitan Area Networks: Link Aggregation.

IEEE Std 802.2™, 1998 Edition [ISO/IEC 8802-2: 1998], Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 2: Logical link control.[6]

IEEE Std 802.3™, IEEE Standard for Ethernet.

IEEE Std 802.11™, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

IEEE Std 802.1AR™, IEEE Standard for Local and Metropolitan Area Networks: Secure Device Identifier.

IETF RFC 2578, STD 58, Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and Waldbusser, S., April 1999.[7]

IETF RFC 2579, STD 58, Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2), McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and Waldbusser, S., April 1999.

[2]DMTF publications are available from the DMTF at https://www.dmtf.org.

[3] Available at https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml.

[4] IEEE publications are available from The Institute of Electrical and Electronics Engineers https://standards.ieee.org.

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

[6] ISO and ISO/IEC documents are available from the International Organization of Standardization (http://www.iso.org) and from the International Electrotechnical Commission (http://www.iec.ch). These documents are also available from the American National Standards Institute (http://ansi.org).

[7]IETF RFCs are available from the Internet Engineering Task Force website at https://www.ietf.org.

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

IETF RFC 2863, The Interfaces Group MIB using SMIv2, McCloghrie, K., and Kastenholz, F., June 2000.

IETF RFC 3394, Advanced Encryption Standard (AES) Key Wrap Algorithm, J. Schaad, and Housley R., September 2002.

IETF RFC 3418, STD 62, Management Information Base (MIB) for the Simple Network Management Protocol (SNMP), Preshun, R., Case, J., McCloghrie, K., Rose, M., Waldbusser, S., December 2002.

IETF RFC 3629, STD 63, UTF-8, a transformation format of ISO 10646, Yergeau, F., November 2003.

IETF RFC 4346, The Transport Layer Security (TLS) Protocol Version 1.3, Diercks, T., Rescorla, E., April 2006.

IETF RFC 4493, The AES-CMAC Algorithm, Song, J.H., Lee, J., and Iwata, T., June 2006.

IETF RFC 5216, The EAP-TLS Authentication Protocol, Simon, D., Aboba, B., and Hurst, R., March 2008.

IETF RFC 5247, Extensible Authentication Protocol (EAP) Key Management Framework, Aboba, B., Simon, D., and Eronen, P., October 2007.

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

IETF RFC 7317, A YANG Data Model for System Management, Bierman A., and Bjorklund M., August 2014.

IETF RFC 7950, The YANG 1.1 Data Modeling Language, Bjorklund, M., editor., August 2016.

IETF RFC 8343, A YANG Data Model for Interface Management, Bjorklund, M., March 2018.

IETF RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3, Rescorla, E., August 2018.

ISO/IEC 18033-3: 2010, Information technology—Security techniques—Encryption algorithms—Part 3:Block ciphers.

NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions, Lily Chen, October 2009.[8]

[8] NIST Special Publication FIPS 800-108 is available at https://csrc.nist.gov/publications/detail/sp/800-108/final.

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.[9,10]

Association Number (AN): A number that is concatenated with a MACsec Secure Channel Identifier to identify a Secure Association.

authentication exchange: A mechanism to verify the identity of an entity by means of information exchange.

NOTE—For example, Extensible Authentication Protocol (EAP) and Simple Authentication and Security Layer (SASL).[11]

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

Authentication Server: An entity that provides an authentication service to an Authenticator. This service determines, from the credentials provided by the Supplicant, whether the Supplicant is authorized to access the services provided by the system in which the Authenticator resides.

NOTE—The Authentication Server function can be co-located with an Authenticator, or it can be accessed remotely via a network to which the Authenticator has access.

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

bounded receive delay: A guarantee that a frame will not be delivered after a known bounded time.

NOTE—In the case of protocols designed to use the MAC Service the receive delay is typically assumed to be less than two seconds.

Bridged Local Area Network: A concatenation of individual IEEE 802 LANs interconnected by MAC Bridges.

NOTE—Unless explicitly specified the use of the word *network* in this Standard refers to a Bridged Local Area Network. The term Bridged Local Area Network is not otherwise abbreviated. The term Local Area Network and the abbreviation LAN are used exclusively to refer to an individual LAN specified by a MAC technology without the inclusion of Bridges. This precise use of terminology within this specification allows a Bridged Local Area Network to be distinguished from an individual LAN that has been bridged to other LANs in the network. In more general usage such precise terminology is not required.

Bridge Port: A Port of an IEEE 802.1Q Bridge.

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

clear: When applied to a Boolean variable, **clear** means to make the value of the variable false. *See also:* **set**.

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

Common Port: an instance of the MAC Internal Sublayer Service used by the SecY to provide transmission and reception of frames for both the controlled and uncontrolled ports.

Controlled Port: The access point used to provide the secure MAC Service to a client of a SecY.

[9]*IEEE Standards Dictionary Online* is available at: http://dictionary.ieee.org. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

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

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

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

a) The transformation from plain text to cipher text and vice versa
b) Synchronized generation of keying material
c) Digital signature computation or validation[12]

cryptographic mode of operation: Also referred to as mode. An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm.[13]

data integrity: A property whereby data has not been altered in an unauthorized manner since it was created, transmitted, or stored.[14]

extended packet number (XPN): A 64-bit packet number (PN) specified in IEEE Std 802.1AE.

frame: MAC protocol data unit (MPDU).

IEEE 802 Local Area Network (LAN): An instance of a LAN technology that provides a MAC Service equivalent to that defined in IEEE Std 802.1AC. IEEE 802 LANs include IEEE Std 802.3 (CSMA/CD) and IEEE Std 802.11 (Wireless).

NOTE—IEEE 802 LANs are also referred to in the text of this standard simply as LANs.

ifIndex: A unique identifying number associated with a logical or physical interface. The term ifIndex is defined and used by SNMP standards.

initialization vector (IV): A vector used in defining the starting point of an encryption process within a cryptographic algorithm.[15]

integrity: *See:* **data integrity**.

Integrity Check Value (ICV): A value that is derived by performing an algorithmic transformation on the data unit for which data integrity services are provided. The ICV is sent with the protected data unit and is recalculated and compared by the receiver to detect data modification.

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

Key Agreement Entity (KaY): A PAE entity responsible for MKA.

key management: The generation, storage, distribution, deletion, archiving, and application of keys in accordance with a security policy.

Key Management Domain (KMD): A string that identifies systems that share cached CAKs.

Layer Management Interface (LMI): The interface between a protocol entity in a system and the system management, providing for the exchange of parameters with other system entities that are not attached to the service access points used and provided by the protocol entity.

MAC Security Entity (SecY): The entity that operates the MAC Security protocol within a system.

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

[13]This and other definitions in this clause have been drawn from Federal Information Processing Standard (FIPS) 300-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001.

[14]This and other definitions in this clause have been drawn from Federal Information Processing Standard (FIPS) 800-57, Recommendation for Key Management, 2005.

[15]This and other definitions in this clause have been drawn from Federal Information Processing Standard (FIPS) 140-2, Security Requirements for Cryptographic Modules, 2001.

MAC Security TAG (SecTAG): A protocol header, comprising a number of octets and beginning with an EtherType, that is prepended to the service data unit supplied by the client of the protocol, and is used to provide security guarantees.

MAC service data unit (MSDU): A sequence of zero or more octets that compose the data to be communicated with a single MAC Service request or indication.

master key: A secret key that is used to derive one or more cryptographic keys that are used directly to protect data transfer.

message authentication: If the message arrives authenticated, the cryptographic guarantee is that the message was not modified in transit and that the message originated from an entity with the proper cryptographic credentials.

mode: *See:* **cryptographic mode of operation.**

multipoint: Involving or potentially involving more than one participant in the role of receiver, or in the role of transmitter, in a single data transfer or set of related data transfers.

network access point: A system, typically incorporating bridging or routing functionality, that comprises one or more network access ports that provides controlled access to a network for one or more systems, usually hosts.

network access port: A point of attachment of a system to a LAN. It can be a physical port, for example, a single LAN MAC attached to a physical LAN segment, or a logical port, for example, an IEEE 802.11 association between a station and an access point.

Network Identity (NID): A UTF-8 (IETF RFC 3629) string identifying an network or network service.

nonce: A non-repeating value, such as a counter, used in key management protocols to thwart replay and other types of attack.

packet number (PN): A monotonically increasing value that is guaranteed unique for each MACsec frame transmitted using a given SA.

Port: A service access point for the MAC Service or MAC Internal Sublayer Service.

port access entity (PAE): The protocol entity associated with a Port. It can support the protocol functionality associated with the Authenticator, the Supplicant, or both.

Port Identifier: A 16-bit identifier that uniquely identifies each of a system's transmit SCs that uses the same MAC address as a component of its SCI.

NOTE—The Port Identifier is not constrained to correspond to any other identifier, index, or port number. There can be more than one SC for a physical port, identifying frames transmitted by separate virtual ports, and more than one SC for a physical or virtual port if that port uses different SCs to transmit frames of different priorities.

protocol data unit (PDU): A unit of data specified in a protocol and consisting of protocol information and, possibly, user data.

real port: A port that is not created on demand, but that can transmit and receive frames for one or more virtual ports.

Salt: A 96-bit secret value communicated by key agreement protocol for use by the protection and verification operations of the IEEE Std 802.1AE GCM-AES-XPN Cipher Suites.

secret key: A cryptographic key used with a secret key cryptographic algorithm that is uniquely associated with one or more entities and should not be made public.[16]

Secure Association (SA): A security relationship that provides security guarantees for frames transmitted from one member of a CA to the others. Each SA is supported by a single secret key, or a single set of keys where the cryptographic operations used to protect one frame require more than one key.

Secure Association Identifier (SAI): An identifier for an SA, comprising the SCI concatenated with the Association Number (AN).

Secure Association Key (SAK): The secret key used by an SA.

Secure Channel (SC): A security relationship used to provide security guarantees for frames transmitted from one member of a CA to the others. An SC is supported by a sequence of SAs thus allowing the periodic use of fresh keys without terminating the relationship.

Secure Channel Identifier (SCI): A unique identifier for a secure channel, comprising a MAC Address and a Port Identifier.

NOTE—Key agreement protocols such as MKA are responsible for ensuring that each SCI used with a given SAK is unique where a Cipher Suite requires that for nonce construction, as does the Default Cipher Suite (14.5 of IEEE Std 802.1AE-2018). SCI uniqueness does not rely on MAC Address allocation procedures.

secure Connectivity Association (CA): A security relationship, established and maintained by key agreement protocols, that comprises a fully connected subset of the service access points in stations attached to a single LAN that are to be supported by MACsec.

secure Connectivity Association Key (CAK): A secret key possessed by members of a given CA.

secure Connectivity Association Key Name (CKN): A text that identifies a CAK.

secured connectivity: Data transfer between two or more Controlled Ports that is protected by MACsec.

NOTE—MACsec requires mutual authentication between communicating peers, so secured connectivity is always authenticated. Connectivity that is not secured could have been preceded by mutual authentication, but is not proof against intrusion.

set: When applied to a Boolean variable, **set** means to make the value of the variable true. *See also:* **clear**.

Short Secure Channel Identifier (SSCI): A 32-bit value that is unique for each SCI within the context of all SecYs using a given SAK.

NOTE—IEEE Std 802.1AEbw-2013 specified the calculation of SSCI and Salt values used by the IEEE Std 802.1AE GCM-AES-XPN Cipher Suites from other MKA values. IEEE Std 802.1Xck-2018 constrained the order of entries in the MKPDU Live Peer List to facilitate that calculation.

Supplicant: An entity at one end of a point-to-point LAN segment that seeks to be authenticated by an Authenticator attached to the other end of that link.

NOTE—The term *Supplicant* is used in this standard in place of the term *peer*, used in other access control-related specifications, as peer is used throughout IEEE 802.1 standards and elsewhere to denote a protocol entity of equal status in a layered protocol model.

system: A device that is attached to a LAN by one or more Ports. Examples of systems include end stations, servers, MAC Bridges, and routers.

unauthenticated connectivity: Data transfer between peers that are not mutually authenticated (such data transfer cannot be secured by MACsec).

Uncontrolled Port: The access point used to provide the insecure MAC Service to a client of a SecY.

[16]FIPS 140-2.

unsecured connectivity: Data transfer that is not protected by MACsec, but has been preceded by mutual authentication between the (supposedly) communicating peers.

virtual port: A MAC Service or Internal Sublayer service access point (IEEE Std 802.1AC) that is created on demand. Virtual ports can be used to provide separate secure connectivity associations over the same LAN.

YANG: A data modeling language, published as IETF RFC 7950.

4. Acronyms and abbreviations

The following abbreviations and acronyms are used in this standard:

AAA	authentication, authorization, and accounting
AES	Advanced Encryption Standard
AN	Association Number
ASF	Alert Standard Format
CA	Secure Connectivity Association
CAK	Secure Connectivity Association Key
CKN	Secure Connectivity Association Key Name
CMAC	Cipher-based Message Authentication Code
CP	PAE Controlled Port state machine
DA	Destination Address
DHCP	Dynamic Host Configuration Protocol
EAP	Extensible Authentication Protocol
EAP-TLS	EAP Transport Layer Security
EAPOL	EAP over LANs
EPON	Ethernet Passive Optical Network
ES	End station
FIPS	Federal Information Processing Standard
ICK	ICV Key
ICV	Integrity Check Value
IP	Internet Protocol
ISS	Internal Sublayer Service
IV	Initialization Vector
KaY	MAC Security Key Agreement Entity
KDF	Key Derivation Function
KEK	Key Encrypting Key
KI	Key Identifier
KMD	Key Management Domain
LAN	IEEE 802 Local Area Network
LLC	Logical Link Control
LLDP	Link Layer Discovery Protocol
LLPN	Lowest acceptable PN for the Latest Key
LMI	Layer Management Interface
LPN	Lowest acceptable PN
MAC	Media Access Control
MI	Member Identifier
MIB	Management Information Base
MKA	MACsec Key Agreement protocol
MKPDU	MACsec Key Agreement Protocol Data Unit
MN	Message Number
MPDU	MAC Protocol Data Unit
MSAP	MAC Service Access Point
MSDU	MAC Service Data Unit
MSK	Master Session Key
NID	Network Identity
NIST	National Institute of Standards and Technology
OLPN	Lowest acceptable PN for the Old Key
OUI	Organizationally Unique Identifier
PAC	Port Access Controller
PACP	Port Access Control Protocol
PAE	Port Access Entity

PBBN	Provider Backbone Bridged Network
PBN	Provider Bridged Network
PDU	Protocol data unit
PN	Packet Number
PSK	pre-shared key
PVID	Port VID
RADIUS	Remote Authentication Dial in User Service
RNG	Random number generator
SA	Secure Association
SAI	Secure Association Identifier
SAK	Secure Association Key
SASL	Simple Authentication and Security Layer
SC	Secure Channel
SCB	Single Copy Broadcast
SCI	Secure Channel Identifier
SecTAG	MAC Security TAG
SecY	MAC Security Entity
SMIv2	Structure of Management Information version 2
SNAP	Subnetwork Access Protocol
SNMP	Simple Network Management Protocol
SSCI	Short SCI
TLV	Type Length Value (a form of encoding, or an item encoded using that encoding)
VID	VLAN Identifier
VLAN	Virtual LAN
WoL	Wake-on-LAN
XPN	Extended Packet Number

5. Conformance

A claim of conformance to this standard is a claim that the behavior of a system meets requirements of this standard for the operation of protocols, management of protocol operation, and provision of service to other systems, as revealed through externally observable behavior of the system.

This clause specifies requirements and options for implementation of system components (5.3) that support port-based network access control, and requirements and options for each of those components (5.4–5.24). Conformance to this standard does not ensure that a system is secure, or that other protocols supported by the system do not provide a way for an attacker to breach that security.

5.1 Requirements terminology

For consistency with existing IEEE and IEEE 802.1 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, 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 (Annex A) reflects 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 implementor 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 word ***allow*** means that the mechanisms specified by this standard are intended to support the indicated use or capabilities, and the word ***capable*** means “is configurable such that …”.

5.2 Protocol Implementation Conformance Statement

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

Where this conformance clause (Clause 5) requires additional information concerning the capabilities of an implementation that information shall also be specified as part of the PICS.

Port-based network access control can be implemented for each of a number of ports of a given system. The ports, and any restrictions placed on individual ports or their total number, shall be specified. If the components or options implemented for different classes of ports differ, a separate PICS shall be completed for each class of port.

The provisions of this standard can require the implementation of one or more MAC Security Entities for each port, as specified by IEEE Std 802.1AE. One or more PICS, as specified by IEEE Std 802.1AE, shall be completed for those ports.

5.3 Conformant systems and system components

This clause (Clause 5) specifies requirements and options for implementation of the following components:

a) Port Access Entity (PAE, see 6.3, Clause 12) (5.4, 5.5, 5.6–5.15, 5.18, 5.19)

b) Port Access Controller (PAC, see 6.4) (5.20)

A port for which conformance to this standard is claimed shall implement the mandatory functions of the PAE (5.4) and the mandatory functionality for at least one of the following PAE functions:

— Supplicant (5.6, 5.7)

— Authenticator (5.8, 5.9)

— MACsec Key Agreement (MKA) (5.10, 5.11)

That port may also implement the mandatory functionality for all or any of the following PAE functions:

— Announcement transmission (5.14, 5.15)

— Announcement reception (5.16, 5.17)

— SNMP access to the PAE MIB (5.18, 5.19)

— Network configuration protocol (e.g., NETCONF) access to the YANG configuration and operational state model of the PAE and PAE system (5.23, 5.24)

A port that implements Authenticator or MKA functionality may also implement the mandatory functionality for the following PAE function:

— Virtual ports (5.12)

A port may implement the optional functionality for any function for which the mandatory functionality has been implemented.

The operation of one or more MAC Security Entities (as specified by IEEE Std 802.1AE) can be required to secure communication for each port. A port that does not implement a SecY shall implement the PAC (5.20).

Any implementation that is claimed to conform to this standard shall not violate the provisions of 5.22 (Prohibitions).

5.4 PAE requirements

A PAE shall

a) Encode, decode, address, and validate EAPOL PDUs as specified in Clause 11.

b) Implement the Logon Process functionality specified in 12.5.

c) Implement the CP state machine as specified in 12.4.

d) Maintain and allow retrieval of the EAPOL frame reception statistics specified in 12.8.1.

e) Maintain and allow retrieval of the EAPOL frame reception diagnostics specified in 12.8.2.

f) Maintain and allow retrieval of the EAPOL frame transmission statistics specified in 12.8.3.

g) Support the system configuration functions specified in 12.9.

A PAE that supports both EAP (Supplicant or Authenticator) and MKA functionality shall

h) Derive a CAK from each EAP MSK, and a corresponding CKN for the corresponding EAP session ID as specified in 6.2.2.

A claim that an implementation conforms to this standard shall specify

i) The group address used to transmit group addressed PDUs. If the implementation can be configured in a way that changes the address used, that configuration shall be specified.

A PAE shall not

j) Use the MACsec protected Controlled Port controlled by the PAE to transmit EAPOL Packet Types other than EAPOL-Announcement (10.2).

5.5 PAE options

This standard does not specify any options for PAE conformance other than those detailed in 5.6 through 5.15.

5.6 Supplicant requirements

A Supplicant shall

a) Use EAP as an authentication protocol, exhibiting external behavior consistent with an implementation of the Supplicant PACP state machine, state machine variables, procedures, and interfaces specified in Clause 8.

b) Not use any EAP method that fails to meet the requirements specified in 8.11.

5.7 Supplicant options

5.7.1 Integration with IEEE Std 802.1AR

An implementation that is claimed to conform to the provisions of this standard for integration with IEEE Std 802.1AR shall

a) Implement the EAP method EAP-TLS as specified in 8.11.2.

5.8 Authenticator requirements

An Authenticator shall

a) Use EAP as an authentication protocol, exhibiting external behavior consistent with an implementation of the Authenticator PACP state machine, state machine variables, procedures, and interfaces specified in Clause 8.

b) Not use any EAP method that fails to meet the requirements specified in 8.11.

c) Allow remote configuration of the reAuthEnabled and reAuthPeriod parameters specified in 8.6 and 8.9.

5.9 Authenticator options

An Authenticator may

a) Maintain and allow retrieval of the Authenticator PAE diagnostic counters specified in 8.10.

5.9.1 Integration with IEEE Std 802.1AR

An implementation that is claimed to conform to the provisions of this standard for integration with IEEE Std 802.1AR shall

a) Implement the EAP method EAP-TLS as specified in 8.11.2.

5.10 MKA requirements

A PAE that supports MKA shall

a) Be capable of maintaining 2 or more simultaneous MKA instances as specified in Clause 9.

b) Specify the maximum number of simultaneous MKA instances it supports.

c) Create, delete, and activate MKA participants as specified in 9.13 and 9.16.

d) Be capable of receiving and using Group CAKs distributed by a Key Server.

e) Meet the requirements for MKA parameter encoding and for MKPDU validation, encoding, and decoding, specified in 11.11.

f) Be capable of using 128 bit CAKs and derived keys as specified in 6.2 and 9.3.

g) Observe the restrictions on the use and disclosure of each CAK and derived keys specified in 6.2 and 9.16.

h) Protect each distributed CAK and SAK by AES Key Wrap, as specified in 9.8.2 and 9.12.1.

i) Include the parameters of EAPOL-Announcements in MKPDUs, as specified in 9.13.

A claim that an implementation conforms to this standard and supports MKA shall specify

j) The MKA Version supported (11.11, Table 11-6, 11.11.3).

5.11 MKA options

A PAE that supports MKA may

a) Be capable of using a 256 bit CAK and derived keys as specified in 6.2 and 9.3.

5.11.1 Support for PSKs

A PAE that supports Pre-Shared Keys (PSKs) shall

a) Support the configuration of PSKs in the CAK Cache as specified in 6.3.3 and 12.6.

5.11.2 Key Server support for Group CAs

A PAE that supports Group CAs as an MKA Key Server

a) Shall be capable of operating as an MKA Key Server (9.5).

b) If support for EAP Authenticator functionality is claimed, shall be capable of:
   1) Distributing a group CAK using MKA with EAP derived pairwise CAKs (6.2, 9.12).

c) If support for PSKs is claimed, shall be capable of:
   1) Distributing a group CAK using an MKA instance with a PSK (6.2, 6.3.3, 9.12).

5.11.3 CAK Cache

A PAE that implements a CAK Cache shall

a) Associate a lifetime with each cached CAK, deleting the CAK when that lifetime has expired as specified in 12.6.

b) Associate a CKN, KMD, NID, and other relevant authorization information with each cached CAK, as specified in 12.6.

c) Cache group CAKs distributed by an MKA Key Server, as specified in 12.6.

d) If support for EAP Authenticator or EAP Supplicant functionality is claimed, shall be capable of:
   1) Caching pairwise CAKs derived from EAP exchanges.

A PAE shall not

e) Cache a CAK derived from an EAP exchange if that is prohibited by 12.6.

f) Distribute a KMD for a Group CAK unless the PAE distributed the CAK when acting as a Key Server as specified in 12.6.

5.11.4 In-service upgrades

A PAE that supports in-service upgrades shall be capable of

a) Suspending MKA operation as specified in 9.18.

b) Communicating the values of the most significant 32 bits of the Lowest Acceptable PN for the Latest Key and the Old Key when any XPN capable Cipher Suite is being used, as specified in 9.18.5.

NOTE—Selection and use of Extended Packet Numbering depends on the implementation of an XPN capable Cipher Suite by each SecY participating in a CA. See IEEE Std 802.1AE.

A PAE that supports in-service upgrades may use additional protocol(s), outside the scope of this specification, to coordinate in-service upgrades as specified in 9.18.6.

5.12 Virtual port requirements

A PAE that supports the use of virtual ports shall

a) Specify the number of virtual ports that can be supported for each real port, or for the system as a whole.

b) Implement MKA functionality, and be capable of operating 2 or more simultaneous MKA instances for each of the specified number of virtual ports.

c) Support each virtual port with a SecY.

d) Create and manage virtual ports as specified in 6.3.6, 9.14, and 12.7.

e) Shall not create or maintain virtual ports if Supplicant functionality is enabled for the real port.

A PAE that supports both virtual ports and Authenticator functionality shall

f) Be capable of supporting at least the same number of simultaneous EAP Exchanges as the number of virtual ports supported.

A PAE implementation that creates virtual ports in a system that bridges frames to and from those ports as specified by IEEE Std 802.1Q shall

g) Support each of those virtual ports as specified by IEEE Std 802.1Q for each Bridge Port, including support for Spanning Tree Protocol (see 7.6).

5.13 Virtual port options

No options are specified for virtual port creation and deletion.

5.14 Announcement transmission requirements

A PAE that transmits EAPOL-Announcements shall

a) Support the use of the null NID and zero or more additional NIDs as specified in Clause 10.

b) Be capable of using the Uncontrolled Port to transmit generic EAPOL-Announcements containing the information for each supported NID, as specified in 10.1 and 10.2.

c) Encode EAPOL-Announcements as specified in 11.12.

d) Rate limit the transmission of announcements as specified in 10.2.

A PAE that transmits EAPOL-Announcements shall not

e) Transmit EAPOL-Announcements through an unsecured Controlled Port (10.2).

5.15 Announcement transmission options

No options are specified for conformance to Announcement transmission functionality.

5.16 Announcement reception requirements

A PAE that listens to EAPOL-Announcements shall

a) Interpret each received EAPOL-Announcement as specified in Clause 11 and Clause 10.

b) Be capable of using a received EAPOL-Announcement to select one or more NIDs, each representing a network or network service, and using the authentication procedure or procedures advertised for that NID subject to the applicable Logon Process controls as specified in Clause 10 and 12.5.

c) Be capable of soliciting an EAPOL-Announcement by transmitting an EAPOL-Announcement-Req as specified in 11.13.

d) Be capable of encoding a Packet Body in an EAPOL-Announcement-Req, including TLVs to select a NID, as specified in 11.13.

e) Be capable of selecting an appropriate credential, for authentication for access to a desired NID, for use by the PAE's Supplicant (if implemented).

f) Be capable of being configured so as to ignore a received EAPOL-Announcement for all NIDs.

A Supplicant that listens to EAPOL-Announcements shall

g) Be capable of encoding a Packet Body in an EAPOL-Start, including TLVs to select a NID, as specified in 11.6.

5.17 Announcement reception options

No options are specified for conformance to Announcement reception functionality.

5.18 Requirements for SNMP access to the PAE MIB

An implementation that is claimed to conform to the provisions of this standard for access to the PAE MIB using SNMP shall

a) Support access to the MIB module specified in Clause 13 using SNMP version v3 or higher.

5.19 Options for SNMP access to the PAE MIB

No options are specified for SNMP access to the PAE MIB.

5.20 PAC requirements

A PAC shall

a) Provide an Uncontrolled Port, for use by the PAE, and a Controlled Port as specified in 6.4.

5.21 System recommendations

Whether the use of a given system or port is appropriate in a given network access control scenario involves considerations that go beyond the scope of this standard, and include details of the port's interface stack, higher layer EAP functionality, and the mechanisms chosen to support unauthenticated connectivity. Clause 7 illustrates use of the components and protocols specified in this standard in typical scenarios. Table 5-1 makes recommendations for the use of this standard, for each of the system roles in each scenario.

5.22 Prohibitions

An implementation that is claimed to conform to this standard shall not

a) Support access to the MIB specified in Clause 13 or any other MIB that provides access to the functionality provided by the implementation using any version of SNMP prior to version 3.

5.23 Requirement for YANG data model of a PAE

An implementation that is claimed to conform to the provisions of this standard for providing a YANG data model for configuration and operational state of a PAE shall

a) Support access to the YANG model specified in Clause 14 using a network configuration protocol (such as NETCONF).

b) Support the configuration of PAEs within a PAE System using the YANG model (described in Clause 14).

c) Support the retrieval of operational state information of PAEs within a PAE System using the YANG model (described in Clause 14).

5.24 Options for YANG data model of a PAE

a) No options are specified for network configuration protocol access to the PAE YANG model.

Table 5-1—System recommendations

Clause 7 scenario and system role	System role	Minimum functionality	Recommended functionality
7.1 Host access with physically secure LANs	Host	Supplicant, PAC	Supplicant, MKA, Listener,[a] PAC, or SecY
	Network access point	Authenticator, PAC	Authenticator, Announcer, MKA, PAC, or SecY
7.2 Infrastructure support with physically secure LANs	Intermediate System	Supplicant or Authenticator or MKA with PSKs, PAC	Supplicant, Authenticator, Announcer,[b] Listener, MKA with PSKs, PAC, or SecY
7.3 Host access with MACsec and point-to-point LANs	Host	Supplicant, MKA, SecY	Supplicant, Listener, MKA with CAK cache, SecY
	Network access point	Authenticator, MKA, SecY	Authenticator, Announcer, MKA with CAK cache, SecY
7.4 Use with MACsec to support infrastructure LANs	Intermediate System	MKA with PSKs, SecY	Supplicant, Authenticator, Announcer, MKA with PSKs and CAK cache, SecY
7.5 Host access with MACsec and a multi-access LAN	Host	Supplicant, MKA, SecY	Supplicant, Listener, MKA with CAK cache, SecY
	Network access point	Authenticator, MKA, Virtual ports, SecY	Authenticator, Announcer, MKA with CAK cache, SecY
7.6 Group host access with MACsec	Host	Supplicant, MKA, SecY	Supplicant, Listener, MKA with CAK cache, SecY
	Network access point	Authenticator, MKA with Key Server support for Group CA, SecY	Authenticator, Announcer, MKA with Key Server support for Group CA and CAK cache, SecY
7.7 Use with MACsec to support virtual shared media infrastructure LANs	Intermediate System	MKA with PSKs, Key Server support for Group CA, CAK cache, SecY	Supplicant, Authenticator, Announcer,[b] Listener, MKA with PSKs, Key Server support for Group CA, CAK cache, SecY

[a]Listener: PAE option of listening to EAPOL announcements.
[b]Announcer and Listener functionality are recommended for 7.2 and 7.7 to facilitate deployment and error detection. Deployed infrastructure devices typically have a single policy preconfigured.

6. Principles of port-based network access control operation

Port-based network access control allows network administrators to restrict the use of IEEE 802 LAN service access points (ports) to secure communication with authenticated and authorized systems. This clause specifies the following:

a) Functions performed by cooperating systems that support port-based network access control (6.1).

b) The cryptographic key hierarchy used to secure communication (6.2).

c) A Port Access Entity (PAE, 6.3, Clause 12) that can

   1) Use EAP to provide mutual authentication;

   2) Support the configuration of pre-shared keys (PSKs) to support mutual authentication;

   3) Support the use of an IEEE Std 802.1AR device identity to support mutual authentication;

   4) Identify a network and key management domain to allow selection of parameters for network use and to support roaming;

   5) Meet the requirements specified in IEEE Std 802.1AE for support of the SecY, including agreeing the cryptographic keys to be used to protect data.

d) A Port Access Controller (PAC), a protocol shim that may be used to enforce access when the SecY or a media specific method for cryptographically securing communication is not used (6.4).

This clause and the other provisions of this standard also specify how the following may be used in conjunction with this standard:

e) The MAC Security Entity (SecY) specified in IEEE Std 802.1AE

f) Link Aggregation as specified in IEEE Std 802.1AX (6.5)

g) IEEE Std 802.11 (6.6)

h) IEEE Std 802.1AR (5.7, 8.11.2)

The system entities and options implemented for a given port can depend on the port-based network access control applications that the system is designed to support. For example, when cryptographic protection of data by a SecY is not required, a PAC can be used. The design of the PAE maximizes interoperability between all combinations, so far as the security requirements of each communicating system and the possibility of physical communication permit.

The mechanisms specified in this standard allow any type of system, including bridges and routers, to authenticate, authorize, and secure communications through one of its ports to any other system attached to the same LAN. Since that other system can also be a bridge or router, authentication and authorization can be required to determine whether each system can trust the other to extend the network and the authentication boundary to its own ports, prior to permitting traffic through controlled ports.

An understanding of architectural concepts common to this and other IEEE 802.1 standards is essential to understanding this standard's specification of port-based network access control. The reader is encouraged to review Clause 7 of IEEE Std 802.1AC-2016 prior to reading the rest of this standard and needs to be familiar with IEEE Std 802.1AE's description of secure LAN communication architecture including the fundamental concept of a Secure Connectivity Association (CA).

6.1 Port-based network access control architecture

The system architecture that supports port-based network access control includes the following:

a) The port, i.e., entities that compose an interface stack supporting a MAC Service access point, through which communication is controlled and secured.

b) The attached LAN, providing the MAC service to the port's client (IEEE Std 802.1AC) and its peers.

c) Mechanisms that define a connectivity association between the port and a peer port or ports.

and, for each port that is a potential peer in secure access controlled communication, a protocol entity or entities that

d) Possess an authentication credential, is within or attached to the same system as the port, and has a secure relationship with the port and its clients.

e) Mutually authenticate peer ports, and whose operation results in

1) Success or failure of the authentication.

2) A token, comprising cryptographic keys and associated data, that can serve as a proof of mutual authentication in subsequent protocols.

3) A binding to authorization data or to a secure channel that can be used to communicate authorization data to the access controlled port and its clients.

f) Communicate authorization data to the port's clients, so that each can permit or deny the use of manageable protocol capabilities appropriately.

g) Use the results of authentication to agree on keys to cryptographically protect communication, creating a secure connectivity association between peer ports.

h) Protect the data transferred within the secure connectivity association.

i) Enforce access control based on the success or failure of the authentication.

Figure 6-1 illustrates the relationships between each of these processes, in an application scenario that uses a AAA server to centralize administration of authentication and authorization.

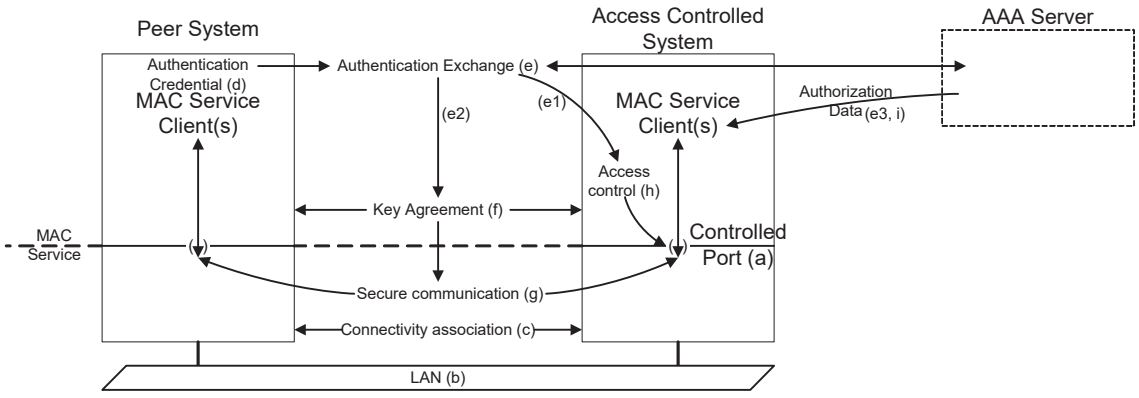


Figure 6-1—Port-based network access control processes

While the protocol entities and options used to implement port-based network access control can vary, most application scenarios (Clause 7) require the majority of functions shown in Figure 6-1 to be present and performed by entities that have the same relationship to each other and to other system components. Figure 6-2 illustrates these with reference to the use of MACsec with authentication and authorization provided by a AAA server, using EAP: the Access Controlled and Peer Systems shown in Figure 6-1 act as the EAP Authenticator and the EAP Supplicant respectively.

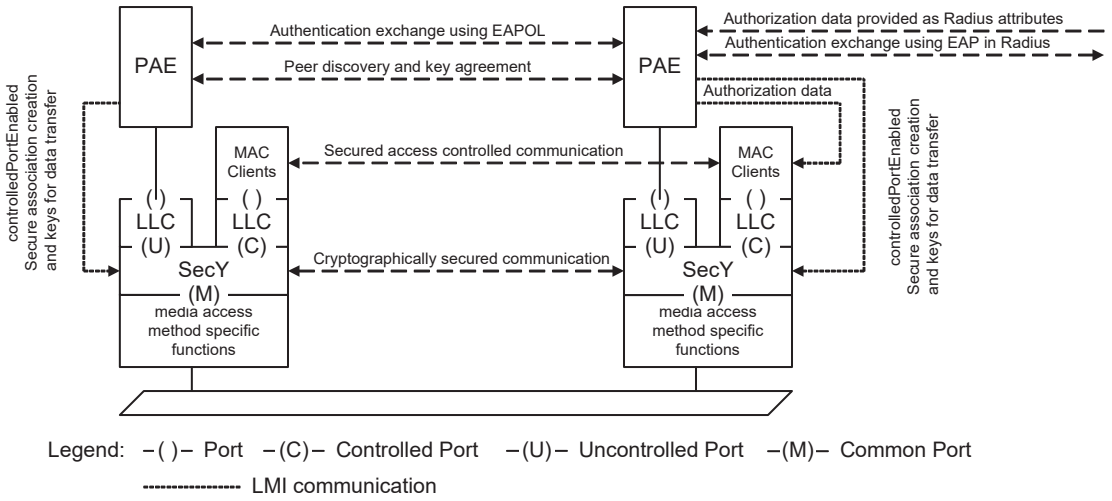


Figure 6-2—Port-based network access control with MACsec

In Figure 6-2, authentication exchanges are supported by the Port Access Entity (PAE, Figure 6.3, Clause 12) using EAP and EAPOL (Clause 11) as specified in this standard (Clause 8). In order to provide a specific example Figure 6-2 references the use of EAP in RADIUS to support authentication exchanges with a AAA Server; this standard does not mandate or constrain the use of methods of transporting EAP other than on a LAN that supports the Controlled Port. Similarly the figure references the communication of authorization data as RADIUS attributes; this standard does not constrain the data supplied or the protocols that convey it. Annex D (informative) provides guidelines for the use of RADIUS with this standard.

The two PAEs use the results of their mutual authentication (see 6.2), and the MACsec Key Agreement protocol (MKA, Clause 9) that in turn uses certain EAPOL PDUs (Clause 11), to communicate securely with each other. The PAEs agree the cryptographic keys to be used by their respective MAC Security Entities (SecY's) that secure communication to and from their Controlled Ports.

For simplicity Figure 6-2 can be taken as depicting a single point-to-point LAN, and the resulting point-to-point CA. A PAE and a SecY can also support multipoint LANs and group CAs. Additionally, a PAE in a network access point supporting a multi-access LAN (7.5, 12.7) can reduce multipoint LAN connectivity to one or more subsets independently secured by SecYs: each SecY supporting a separate virtual port.

A given system can be attached to one of many LANs, with potential peers providing access to many different networks or network services. A PAE for a system that provides network access can announce the identity or identities of those network(s). A given system can also be moved (roam) from one network to another before being reconnected (roaming) to the first. In that case communication can be re-established quickly if both communicating PAEs have cached the results of a prior mutual authentication—peer discovery and key agreement can be used to confirm the authentication and use it to agree fresh keys to protect data transfer, while a fresh authentication exchange with the AAA server is in progress.

6.2 Key hierarchy

The root key in the MACsec Key Agreement key hierarchy is the Connectivity Association Key (CAK), and is identified by a CAK Name (CKN). MKA (Clause 9) does not use the CAK directly but derives two further keys from the CAK using the AES cipher (ISO/IEC 18033-3) in CMAC mode (9.3). These are the ICV Key (ICK) used to verify the integrity of MPDUs and to prove that the transmitter of the MKPDU possesses the CAK, and the Key Encrypting Key (KEK) used by Key Server, elected by MKA, to transport a succession of SAKs, for use by MACsec, to the other member(s) of a CA. See Figure 6-3.

A CAK can be obtained in several ways. It can be a pre-shared key (PSK) configured as specified in 6.3.3, 12.1, and 12.5. The PSK is identified as being a pairwise CAK or as a group CAK. Alternatively key management can be automated. This standard specifies the use of EAP for automatic CAK management through mutual authentication and key derivation of a pairwise CAK.

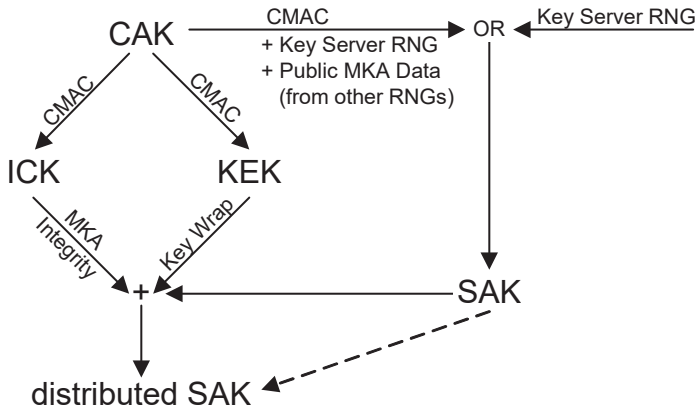


Figure 6-3—MKA key hierarchy

The ICK and KEK derived from a CAK can be used to distribute SAKs to systems that possess that CAK. A number of pairwise CAKs, each possessed only by the Key Server and one other system, can also be used by the Key Server to derive ICK, KEK tuples to distribute a further, different, group CAK. See Figure 6-4.

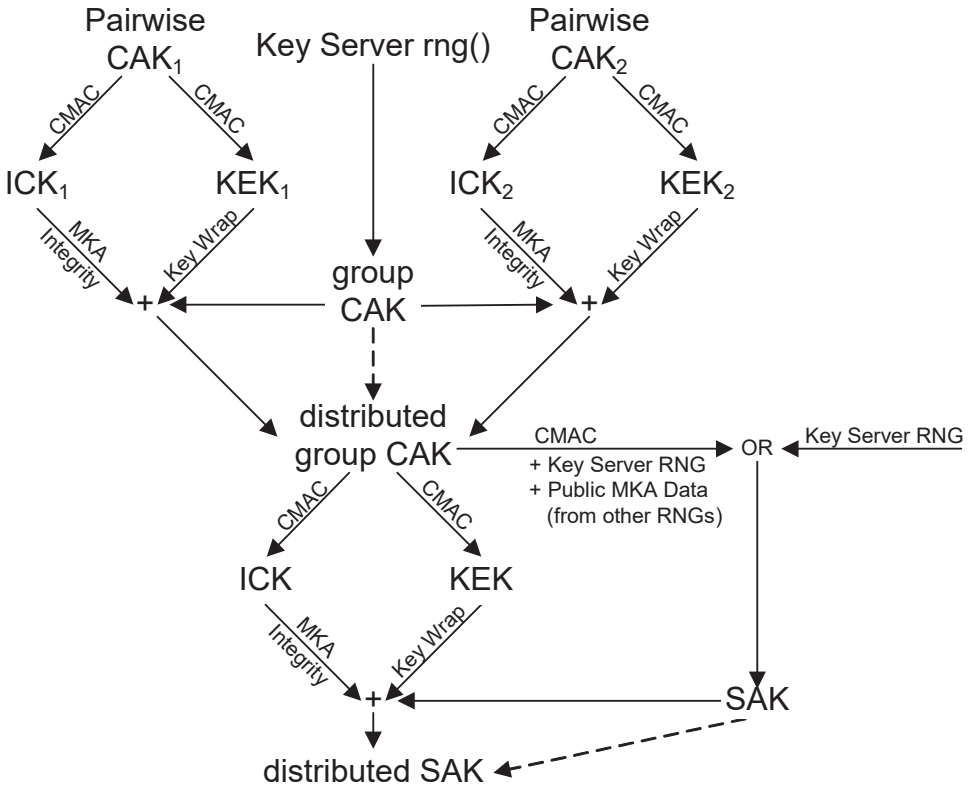


Figure 6-4—Use of pairwise CAKs to distribute group SAKs

Each of the communicating systems use this group CAK to derive a further ICK and KEK, that are then used by the Key Server to transport a succession of SAKs to the group. A group CAK can be used to distribute a further group CAK, but systems can be configured only to accept CAK distributed using a pairwise CAK. The ICK and KEK derived from a CAK shall not be used to distribute that CAK, and in particular a pairwise CAK derived using EAP shall not be distributed for use as a group CAK.

SAKs should be generated by MKA using the CAK with a random number generated specifically for that SAK and contributions from each of the participants in the CAK (9.8.1). These contributions guard against weakness in the Key Server's RNG but do not relax the RNG conformance requirement. Alternatively, SAKs may be generated directly by the Key Server's strong random number generator (RNG, 9.2.1).

6.2.1 Key derivation function (KDF)

This standard uses a KDF that is compatible with the counter mode KDF described in the NIST Special Publication 800-108. The KDF uses a pseudorandom function (PRF) which shall be AES-CMAC-128 when the derivation key is 128 bits, and AES-CMAC-256 when the derivation key is 256 bits.

The KDF is described as follows:

Output← KDF (Key, Label, Context, Length) where

Input: Key, a key derivation key of 128 or 256 bits

Label, a string identifying the purpose of the keys derived using this KDF

Context, a bit string that provides context to identify the derived key

Length, the length of the output in bits encoded in two octets with the most significant octet first

Output: a Length-bit derived value

Fixed values:

h, the length of the output of the PRF in bits

r, denoting the length of the binary representation of the counter i

iterations ← (Length + (h–1))/h

if iterations > 2^r–1, **then** indicate an error and stop.

result ← ""

do i = 1 to iterations

result ← result | PRF(Key, i | Label | 0x00 | Context | Length)

od

return first Length bits of result, and securely delete all unused bits

NOTE —'|', denotes concatenation of the octet strings; i and 0x00 each comprise a single octet; and Label, Context, and Length each comprise an integral number of octets. The hexadecimal representations of each of the Labels specified by this standard for use with this KDF are specified in Annex G.

6.2.2 Using EAP for CAK key derivation

A pairwise CAK is derived directly from the EAP MSK using the following transform:

CAK = KDF(Key, Label, mac1 | mac2, CAKlength)

where

Key = MSK[0-15] for a 128 bit CAK, MSK[0-31] for a 256 bit CAK

Label = "IEEE8021 EAP CAK"

mac1 = the lesser of the two source MAC addresses used in the EAPOL-EAP exchange (11.1.2)

mac2 = the greater of the two source MAC addresses used in the EAPOL-EAP exchange

CAKlength = two octets representing an integer value (128 for a 128 bit CAK, 256 for a 256 bit CAK) with the most significant octet first

NOTE 1—MSK[0-15] refers to the first 16 octets of the MSK, MSK[0-31] to the first 32. IETF RFC 3748 [B14] requires EAP methods that produce an MSK to create an MSK of 64 bytes, so there will be sufficient keying material for the KDF.

NOTE 2—The source MAC addresses, mac1 and mac2, are those that supported the EAPOL-EAP exchange. While including these addresses in the derivation ensures that the CAK and CKN differ if the MSK[0-15] chance to be the same for authentication instances using different ports, they do not define or restrict the scope of the CAK and CKN, and do not identify the PAEs that anticipated in the authentication exchange. The scope of the CAK and CKN is identified by the KMD (see 6.2.3, 6.3.5).

A 16 octet CKN is derived from the EAP session ID using the following transform:

CKN = KDF(Key, Label, ID | mac1| mac2, CKNlength)

where

Key = MSK[0-15] for a CKN naming a 128 bit CAK, MSK[0-31] for naming a 256 bit CAK

ID = EAP-Session-ID

Label = "IEEE8021 EAP CKN"

mac1 = the lesser of the two source MAC addresses used in the EAPOL-EAP exchange (11.1.2)

mac2 = the greater of the two source MAC addresses used in the EAPOL-EAP exchange

CKNlength = two octets representing an integer value (128) with the most significant octet first

NOTE 3—EAP-Session-IDs are defined by IETF RFC 5247. Each Session-ID comprises a number of concatenated fields: octets corresponding to fields earlier in the concatenation (shown to the left in text) are encoded earlier in PDUs and multi-octet fields that represent numbers or can be numerically compared are encoded as specified by this standard (in network byte order).

In each case the Label is a UTF-8 string, without a null or other termination, exactly 16 bytes in length (the quotes shown above are not part of the string) and exactly one space (a single octet with the value 0x20) separates ‘8021’ and ‘EAP ‘, and one space separates ‘EAP’ and ‘CAK’, and ‘EAP’ and ‘CKN’. The Label’s length is chosen to make the concatenated inputs to the PRF within the KDF for the CAK exactly 32 octets.

For historical reasons MAC addresses, and the 24-bit OUIs assigned so that the assignee could derive a number of MAC addresses, were defined as bit-strings. mac1 and mac2 above are sequences of 6 octets, with the value of first octet derived from the first 8 bits of the 48-bit MAC address string, the second octet from second set of eight bits and so on, each set of 8 bits being taken to represent a binary number with its first bit being the least significant. The value of each octet in the sequence corresponds to that naturally associated with the Hexadecimal Representation of the LAN MAC Address as specified in IEEE Std 802, and the order of the octets corresponds to the left to right order in that representation. The first bit of the address string is the Individual/Group address, and the bit order of the address is that originally used for transmission on IEEE 802.3 media. The octet sequence defined to represent the MAC address is used in a

number of other IEEE 802.1 standards, including IEEE Std 802.1Q and the value and representation for an address is commonly known as the ‘canonical format’. Users of this specification are warned that the long running ‘endian’ disputes about bit ordering and bit significance mean that some seemingly authoritative and generally available tutorials do reverse the bits of the 48-bit address string in eight bit groups, in order to show the most significant bit to the left (first according to some familiar conventions), though most now retain the bit significance necessary to this specification.

When two MAC addresses are compared, each is treated as an unsigned binary number, following the convention (used throughout IEEE 802.1 standards) that binary numbers are encoded in octet strings with the most significant octets first. The greater of two addresses is that with the greater numeric value.

6.2.3 CAK caching and scope

A CAK and its associated CKN may be cached for later use. The scope of use of a CAK is identified as a Key Management Domain (KMD). While the initial derivation of the CAK and the CKN include values particular to the PAEs that participate in an instance of authentication protocol (e.g., the MAC addresses of the associated ports) and to that protocol instance (e.g., the EAP-MSK), the CAK can be used as a proof of authentication with any port controlled by a PAE that participates in the same KMD (see 6.3.5). Neither the CAK or the CKN is recalculated following their initial derivation.

The cached CAK is associated with its CKN and any other contextual parameters associated with its use, such as the NID and authorization data. It can be inappropriate to cache a CAK since authorizations or other important contextual information may be dynamic in nature.

6.2.4 Algorithm agility

To accommodate future developments in cryptography, MKA provides an explicit Algorithm Agility parameter (9.3.3, 11.11.2, Figure 11-8). The Algorithm Agility parameter identifies the following:

a) How the ICV is derived from the CAK and the data of a given MKPDU; and

b) How a CAK is derived from the parameters available to the participants in an EAP exchange.

Knowledge of item a) is required for MKA’s use of CAKs in general, while both item a) and item b) are required for the validation of MKPDUs that are protected using the results of an EAP exchange. MKA instances that transmit MKPDUs with different values of the Algorithm Agility parameter could use a different KDF (see 6.2.1) to derive the ICK (9.3.3), could specify the computation of the ICV in a different way, and could specify a different way for the participants in an EAP exchange to agree on a CAK (6.2.2). The flexibility provided allows a wide range of future challenges to be addressed, but the need for substantial analysis of any proposed alternate to the provisions of this standard in these areas means that no claim of conformance is facilitated currently for any such alternate. A future revision of this standard could provide such a claim, if needed to address cryptographic developments.

MKA’s Algorithm Agility parameter does not identify the KDF used to derive the KEK, or the Key Wrap (see 6.2). They are identified by the MKA parameter set type used to encode the Key Wrap. MKA parameter sets 4 and 5 (Figure 11-11, Figure 11-12, Figure 11-13) use a particular KDF (6.2.1, 9.3.3) and Key Wrap (AES Key Wrap, 9.8.2, 9.12.1) by definition. A different Key Wrap, or a Key Wrap using a differently derived KEK, could be introduced by defining a further parameter set type while still allowing MKA participants to communicate by using MKPDUs with a familiar Algorithm Agility parameter value—thus allowing negotiation or fallback to known parameter set types.

6.3 Port Access Entity (PAE)

The operation of the PAE is specified in detail in Clause 12. This clause (6.3) provides an introduction to allow the principles of port-based network access control as a whole to be understood, providing the context for operation of the protocols that the PAE uses (as specified in detail in Clause 8, Clause 9, Clause 10, and Clause 11).

6.3.1 Authentication exchanges

Each PAE may operate EAP (IETF RFC 3748 [B14]). The Uncontrolled Port provided by the SecY or PAC (6.4) is used to transmit EAP frames over the LAN (EAPOL) encoded as specified in Clause 11, as part of the Port Access Control Protocol (PACP) specified in Clause 8. PACP initiates authentication attempts, retries initial authentication (if necessary) to guard against frame loss when the two communicating Ports are enabled at different times, provides periodic reauthentication, and terminates authentication on request, in addition to transparently conveying EAP frames between the EAP PAEs.

In any given authentication exchange, each PAE adopts one of two distinct EAP specified roles:

a) Authenticator

b) Supplicant

NOTE 1—A given PAE can participate in one authentication exchange as a Supplicant, and in another as an Authenticator. See Figure 7-10.

EAP specifies a further system role:

c) Authentication Server

All three roles are necessary to an EAP authentication exchange. The Authenticator and an Authentication Server can be co-located within the same system, allowing that system to perform the authentication function without the need for communication with an external server. However most port-based network access control applications use a separate Authentication Server, to allow centralized administration of authorized parties and their credentials for Authenticators throughout a network. Protocol exchanges between the Authenticator PAE and the Authentication Server (if the server is not co-located with the Authenticator PAE) can be conducted via one or more of the system's Controlled or Uncontrolled Ports.

NOTE 2—Communication between Authenticator and Authentication Server is outside the scope of this standard, but typically uses an authentication protocol carried over appropriate higher layer protocols; e.g., EAP in RADIUS. Hence, the Authentication Server can be located outside of the confines of the LAN that supports the protocol exchanges between Supplicant and Authenticator, and the communication to the server need not be subject to the authentication state of the systems concerned. If a Controlled Port is used to achieve communication with the Authentication Server, protocol exchanges can only take place if the Controlled Port is in the authorized state.

The asymmetry in some port-based network access control applications defines appropriate system roles, for example a mobile personal computer that attaches to a network is a Supplicant and the port that supports its attachment an Authenticator. Other applications, e.g. securing the connection between two bridges in the core of a network, lack this natural asymmetry. A system may implement and enable both Authenticator and Supplicant state machines to support such applications, with the attendant possibility that two separate authentication exchanges will complete. The PAE uses MKA (see 6.3.2) to ensure that communicating peers agree on the use of authentication results: each of the cryptographic keys that it uses (and the authorization data bound to that key) is derived from the result of a single exchange. By itself EAP (supported by PACP) lacks the capability to agree that the results of a particular exchange are to be used, or indeed to confirm that the communicating peers both believe the same exchange has completed, and should not be used without MKA in applications where a single port requires simultaneous Authenticator and Supplicant functionality. However, if MACsec is not used further keys are not required, so communication can and should proceed if either the Supplicant or the Authenticator authenticates successfully and MKA is not available.

NOTE 3—IEEE Std 802.1X-2004 Edition provided explicit control over the combination of authentication results by management of Supplicant Access Control With Authenticator. The use of this variable is deprecated, but not prohibited.

6.3.2 Key agreement

Each PAE may operate the MACsec Key Agreement (MKA) protocol (Clause 9). The Uncontrolled Port provided by the SecY or PAC is used to transmit and receive MKPDUs (11.11) that are conveyed by EAPOL PDUs, distinguished from those used by PACP by their EAPOL Packet Type (11.3.2, Table 11-3).

A PAE can operate multiple instances of MKA. Each instance is protected by a distinct Secure Connectivity Association Key (CAK) that allows each PAE to ensure that information for a given MKA instance is only accepted from other PAEs that also possess that CAK, thus identifying themselves as members or potential members of the same CA. Each CAK is a result of an EAP authentication exchange (6.2), or is configured in two or more PAEs as a pre-shared key (PSK, 6.3.3), or is distributed by a separate MKA instance that uses the results of an authentication exchange. The last of these methods allows a key server's PAE to use a number of authentication exchanges, each with one other PAE, to create a group CA (with a group CAK) that can continue to secure communication in the absence of any of its members (including the key server).

MKPDUs are transmitted using a group MAC destination address, allowing the members of a CA to find each other without prior configuration. MKA can confirm an immediately prior or past authentication, and can be used in conjunction with a PAC (which does not use cryptographic keys itself) to verify connectivity. When used with a SecY, the PAE also uses MKA to distribute SAKs and to assign these to SAs.

6.3.3 Pre-shared keys

A PAE may support the use of one or more pre-shared CAKs (PSKs). An instance of MKA (see Clause 9) operates for each pre-shared CAK that is administratively configured as active, just as if that CAK had been provided (see 6.3.2) by a PAE using EAP (see Clause 8). A PSK can be a pairwise or group CAK.

A pre-shared CAK may be created by a management request with the following parameters:

a) enabled, True or False.
b) CKN, a name for the CAK with the format specified in 9.3.1.
c) The CAK value.

The CAK Name is required to be different from that for existing keys, and can be used to identify the key in subsequent management operations.

If provision of a pre-shared CAK is permitted, use of the pre-shared key can also be enabled or disabled by management request, the CAK name read, or the parameters for the key deleted entirely. The CAK value shall never be returned in response to a management request.

6.3.4 Interoperability and connectivity

Systems incorporating a given port-based network access control implementation can be used in a number of different application scenarios (Clause 7 provides examples) without individual pre-configuration, including attaching to LANs where the other systems do not implement PACP or MKA. A given system can be moved from LAN to LAN, and its current or potential peers added, removed, or reconfigured.

The PAE's use of EAP, PACP, and MKA can be managed. Authenticated and secured connectivity is usually preferred, but the PAE can be configured to permit unauthenticated or unsecured connectivity if required. To minimize any delay that could occur while PACP and MKA conclude that there are no port-based network access control capable peers attached to the LAN, unauthenticated connectivity can be provided until authenticated and secured connectivity is available.

Once secured connectivity is established the PAE manages reauthentication using EAP (if required) and concurrent MKA instances, so that the CAKs that result from successive authentication exchanges provide an uninterrupted supply of SAKs for the SecY's use.

Higher layer protocols that transport EAP can supply policy controls that are outside the scope of this standard but are required by management for the authorization associated with the port. Such policy controls can also be associated with a PSK. The PAE sets controlledPortEnabled for the PAC or SecY if and only if any policy controls, required by management for the authorization associated with the port but outside the scope of this standard, have been applied by the Controlled Port's clients.

NOTE—Where EAP in RADIUS is used for authentication, the RADIUS protocol can supply authorization data including, for example, VLAN configuration of the port (see 7.1.3, IEEE Std 802.1Q, IETF RFC 4675 [B17]).

Changes in the nature of the Controlled Port's connectivity (from one of unauthenticated, unsecured, or secured, to another) and changes in secured connectivity (changes in the CA), are accompanied by controlPortEnabled transitions, and are seen by clients of the Controlled Port as interruptions in connectivity. These interruptions (signaled by changes in MAC_Operational for the port) should be used to restart client protocol state machines that take exceptional or accelerated action when connectivity is first available. For example, a network access point that provides unauthenticated connectivity to a host can provide that host access to a different IP subnet once it has been authenticated, and it is appropriate to reinitialize DHCP so the host can obtain a fresh IP address.

6.3.5 Network announcements, identity, authentication requirements, and status

A PAE may announce the Network Identity (NID), a string identifying a network, of the network(s) for which it controls access, and can solicit announcements from the PAEs of other systems attached to the same LAN. A Supplicant PAE can associate a NID with a configuration profile containing the parameters (e.g., selected credentials for use with EAP) necessary to access that network successfully, and can present NID information during the authentication information exchange to influence authorization processes.

An announcement can state whether the sending PAE supports open or restricted unauthenticated access to that network, PACP and or MKA for authenticated access, and MACsec for secured access, and whether authentication is supported by a higher layer protocol (such as WebAuth). The announcement can also provide the MAC_Operational status of the Controlled Port and the status of the selected NID, to allow a Supplicant to distinguish between the immediate availability of restricted access (or higher layer authentication) and its potential use after failure of an authentication attempt.

Announcements can be conveyed in EAPOL PDUs. EAPOL-Announcements can be solicited by other EAPOL PDUs. Announcements can also be conveyed securely by MKA, so that the recipient can confirm that the actions taken on the unsecured EAPOL-Announcement were appropriate.

PAEs that announce and act on announced information can shorten the time taken for a Supplicant to gain network access. A PAE can also announce its Key Management Domain (KMD), a string that identifies one or more systems that share cached CAKs. This announcement allows a peer PAE to select from amongst its cached CAKs, and re-establish authenticated connectivity before or in the absence of authentication using EAP. Such a PAE can move (roam) from one LAN to another, minimizing the time taken to connect to each.

NOTE—While a Key Management Domain can comprise more than one system, how a number of systems hold a CAK in common or convey it to the particular system that requires it to support roaming is outside the scope of this standard.

6.3.6 Multi-access LANs

A network access point may instantiate multiple virtual ports, each including a SecY, to support individually secured point-to-point connections to each of a number of other stations attached to the same shared-media individual LAN. See Figure 6-5. A network access point providing such virtual ports can support multiple distinct and simultaneous EAP exchanges with PACP, distinguishing between frames to and from each of the other stations by using the station's individual MAC address in the destination address field (on transmission) and from the source address field (on receipt).

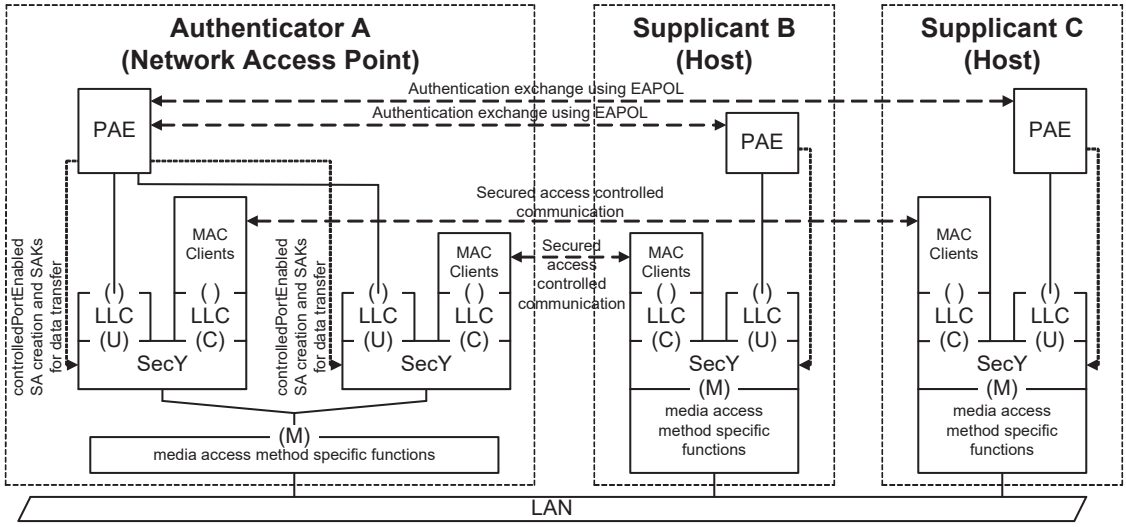


Figure 6-5—Network access control with MACsec and a multi-access LAN

NOTE—The addressing (individual or group) for each EAPOL Packet Type is specified in Table 11-4. For compatibility with prior revisions of this standard, Supplicants attached to IEEE 802.3 networks use only group addresses in the destination address field of EAPOL frames supporting PACP.

A PAE that supports virtual ports shall also be capable of operating individual MKA instances for each virtual port. MKA frames include the CKN, which is sufficient for each station to associate each with the virtual port of potential interest, or to discard it. Other group addressed EAPOL frames, i.e., those with the individual MAC address of a network access point as their source address, apply equally to all stations.

6.4 Port Access Controller (PAC)

The PAC is a protocol-less shim (IEEE Std 802.1AC) that provides control over frame transmission and reception by clients attached to its Controlled Port and uses the MAC Service provided by a Common Port. The access control decision is made by the PAE, typically taking into account the success or failure of mutual authentication and authorization of the PAE's peer(s), and is communicated by the PAE using the LMI to set the PAC's controlledPortEnabled variable. The PAE itself attaches to an Uncontrolled Port, provided by the PAC to support the authentication exchange prior to authorizing use of the Controlled Port. See Figure 6-6. Either or both of the SecY's in Figure 6-2 could be replaced with a PAC if cryptographically secured communication between the two systems were not required. A SecY can be configured, using the management controls specified in IEEE Std 802.1AE, to behave exactly like a PAC—thus facilitating interoperability when only one of the communicating systems implements MACsec.

NOTE—IEEE Std 802.1X-2004 and earlier editions of this standard specified use of the Controlled Port without instantiating an entity that could realize the necessary functionality within an interface stack. The PAC formalizes that description, allows it to be applied to complex interface stacks, and clarifies the relationship with MACsec.

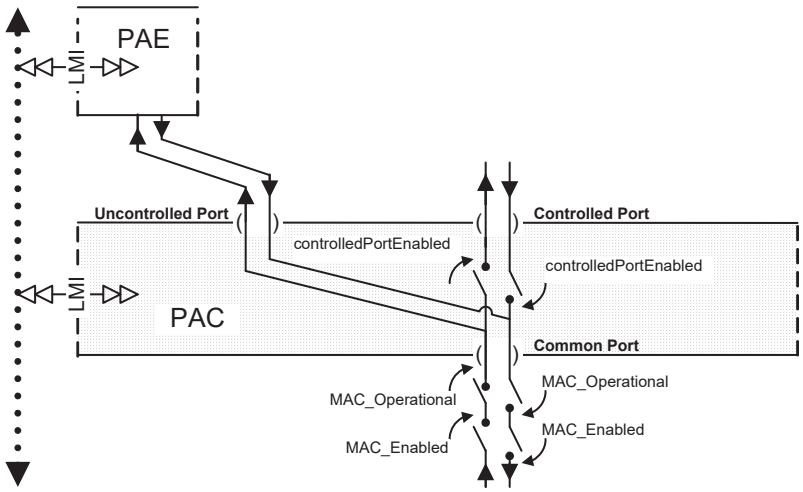


Figure 6-6—Port Access Controller

6.4.1 Uncontrolled Port transmission and reception

Each transmit request from the Uncontrolled Port results in a single and identical transmit request at the Common Port. Each receive indication from the Common Port results in a single and identical receive indication at the Uncontrolled Port in addition to any receive indication at the Controlled Port permitted by authorization and administrative controls (6.4.2).

NOTE 1—This specification most clearly sets out the resulting behavior of a conforming implementation. Real implementations can implement the behavior in any way that yields the same externally visible behavior. There need be no implementation burden corresponding to duplication of a received frame if either of the clients of the Uncontrolled or Controlled Port will discard the frame, e.g., following examination of its destination address or Ethertype.

MAC_Operational for the Uncontrolled Port is set *True* if and only if it is *True* for the Common Port.

NOTE 2—If MAC_Operational is *False*, transmit requests are not accepted and receive indications do not occur. The MAC_Enabled status parameter reflects the result of administrative controls applied to the components that compose a port interface stack, in general if MAC_Enabled is *False*, MAC_Operational will also be *False*.

6.4.2 Controlled Port transmission and reception

Each transmit request from the client of the PAC's Controlled Port results in a corresponding transmit request at the Common Port, and each receive indication from the Common Port results in a corresponding receive indication at the Controlled Port, without omission or duplication, and with identical parameters in each case, if and only if controlledPortEnabled is set.

MAC_Operational (IEEE Std 802.1AC) for the Controlled Port is set *True* if and only if MAC_Operational for the Common Port is *True* and controlledPortEnabled is set.

NOTE—Previous revisions of this standard specified an AdminControlledDirections parameter. This parameter is no longer supported—it did not satisfy its stated goals fully and is not applicable where access is cryptographically protected. The desired functionality was for unauthenticated systems to be able to see startup and initialization traffic (e.g., Wake-on-LAN) functions. This revision of this standard provides explicit support for such functions—see the application-specific discussions of connectivity to unauthenticated systems in 7.5.3, 7.4.3, and elsewhere in Clause 7.

6.4.3 PAC management

The PAC management process controls, monitors, and reports on the operation of the PAC, providing access to operational controls and statistics for network management and the PAE through the LMI.

The following status parameters (IEEE Std 802.1AC) for the Uncontrolled Port and the Controlled Port are provided (separately) to the user(s) of those ports, and can be read by management:

— MAC_Enabled
— MAC_Operational
— operPointToPointMAC
— adminPointToPointMAC

The value of MAC_Enabled for the Uncontrolled Port is always True, the value of MAC_Enabled for the Controlled Port is the same as that of the controlledPortEnabled parameter set by the PAE.

The following statistics are provided for the Controlled Port and (separately) for the Uncontrolled Port to support IETF RFC 2863 interface MIB Counters:

— ifInOctets
— ifInUcastPkts, ifInMulticastPkts, and ifInBroadcastPkts
— ifInDiscards
— ifInErrors
— ifOutOctets
— ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts
— ifOutErrors

The ifInDiscards, ifInErrors, and ifOutErrors counts are zero, as the operation of the Controlled Port and the Uncontrolled Port provides no error checking or occasion to discard packets, beyond that provided by its users or by the entity supporting the Common Port. Discards while the controlledPortEnabled is false are not considered as error (if necessary the Common Port and Controlled Port statistics can be compared).

The values of ifInOctets, ifInUcastPkts, ifInMulticastPkts, and ifInBroadcastPkts for the Uncontrolled Port are identical to those for Common Port, and are not separately recorded, but can be made available to management for consistency of presentation. The values of ifOutOctets, ifOutUcastPkts, ifOutMulticastPkts, and ifOutBroadcastPkts for the Uncontrolled Port can be obtained by subtracting the Controlled Port values of those statistics from those recorded for the Common Port.

6.5 Link aggregation

Whenever access controlled ports can be aggregated (see IEEE Std 802.1AX), a PAE associated with each port authenticates each individually, so that authentication and authorization are not compromised by an aggregation that includes unauthenticated connectivity. Authorized ports are aggregated only with others providing connectivity to the same authenticated system.

NOTE—Conformance to this standard requires the use of protocols that provide mutual authentication, so two link aggregation capable devices that communicate securely will both be aware of the authentication status of aggregatable ports. The requirement for connectivity to the same authenticated system supplements the system identities advertised in aggregation control protocols.

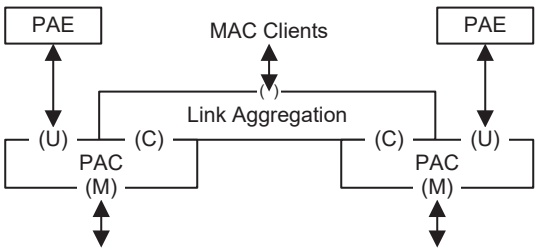


Figure 6-7—PACs and Link Aggregation in an interface stack

Figure 6-7 and Figure 6-8 illustrate the interface stack when access control for each port is provided by the PAC and by the SecY.

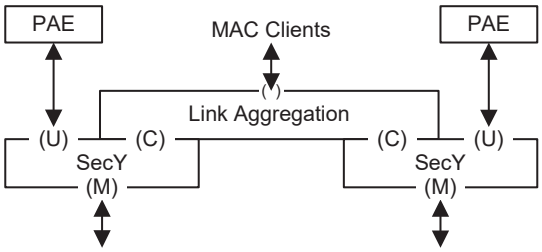


Figure 6-8—SecYs and Link Aggregation in an interface stack

6.6 Use of this standard by IEEE Std 802.11

IEEE Std 802.11 specifies media-dependent cryptographic methods to protect data transmitted using the IEEE 802.11 MAC over wireless networks. Conceptually these cryptographic methods can be considered as playing the same role within systems and interface stacks as a MAC Security Entity. In addition IEEE Std 802.11 specifies media dependent key agreement, key distribution, network discovery, network selection, and roaming protocols. It is not the purpose of this standard to provide alternatives for the IEEE Std 802.11 specified functionality in IEEE 802.11 wireless networks.

IEEE Std 802.11 specifies the use of PACP and EAPOL, as defined in this standard, to authenticate using EAP (see Clause 8, Clause 11) and PAEs that support IEEE 802.11 ports implement PACP, though not the other optional functionality of the PAE.

7. Port-based network access control applications

Port-based network access control requirements and the use of protocol entities to instantiate the port-based network access control architecture (6.1) vary by application. This clause describes the use of port-based network access control in a number of applications, including the following:

a) Host access using individual, physically secure, point-to-point LANs (7.1)
b) Infrastructure support with physically secure, point-to-point LANs (7.2)
c) Host access using MACsec and point-to-point (7.3) or multi-access (7.5) LANs
d) Infrastructure LANs using MACsec (7.4)
e) Group host access using MACsec (7.6)
f) Virtual shared media infrastructure LANs using MACsec (7.7)

This list of applications is not exhaustive, nor is the characterization of each application definitive, but illustrates the need for the conformance options (Clause 5). Requirements and options attempt to maximize interoperability between implementations with the minimum of application specific configuration and knowledge, other than that required strictly to provide security.

A full description of the operation of port-based network access control requires consideration of the system context, so this clause describes how each application can be supported by the ports of VLAN-aware Bridge (IEEE Std 802.1Q). Similar considerations apply to the system configuration of the ports of a router or a system providing bridge router functionality, where each VLAN is associated with a routed subnet. Clause 11 of IEEE Std 802.1AE-2018 describes the use of MACsec in additional system configurations.

In many secure networks, connectivity to unauthenticated systems at the edge of the network is still required. Such unsecured but controlled and limited connectivity allows (for example) newly attached systems to register or subscribe to centrally managed services, or supports management of attached systems that are in a partially powered down state and cannot be expected to maintain their participation in authentication and key agreement protocols, while continuing to protect the operation of the secured network. The basic mechanisms employed are introduced in 7.1.3.

7.1 Host access with physically secure LANs

An important and widespread use of port-based network access control is securing access to a network from a personal computer assumed to be under the direct control of an authorized user. See Figure 7-1.

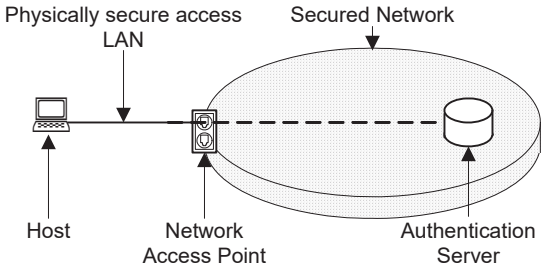


Figure 7-1—Network access control with a physically secure point-to-point LAN

In this application the emphasis is often on protecting the network against theft of service by unauthorized parties. Such protection necessarily involves ensuring that the communication to and from the personal computer is also authenticated, to prevent an attacker hijacking its resources, and gaining unauthorized access to the network indirectly. This standard therefore mandates the use of mutual authentication in all its applications, even if the focus is on protecting only one of the communicating parties.

7.1.1 Assumptions and requirements

The port-based network access control application described in this subclause (7.1) is characterized by the following assumptions and requirements:

a) The network connection is intended to support access by just one system.

b) The host's user or administrator can verify the security of the physical link to the network independently of the mechanisms provided by this standard (probably by visual inspection).

c) There is a clear physical distinction between the network equipment supporting access, on the one hand, and the system accessing the network on the other.

d) Access is only to be provided when the network itself is completely, or at least mostly, operational.

e) If AAA resources for the network are centralized, or separated from the equipment that immediately controls access, the access should not be provided when those resources are inaccessible (e.g., when charging for access is not possible).

f) The authentication credentials presented by the host can correspond to a user or to machine identity.

g) The host, possibly prompted by its user, has to be able to signal that access is no longer desired.

h) If authentication credentials are user-based, periodic reauthentication can be required to check that the user has not logged off the host (the local actions taken in 'logging off' are assumed to make cached credentials or keys inaccessible without the user's cooperation).

i) Reauthentication is required if the access connectivity is disrupted or fails, or if there is a failure within the network.

j) Brief interruptions in communication while reauthentication takes place can be tolerated.

7.1.2 System configuration and operation

The processes and entities that support this application are illustrated in Figure 7-2. The host that is seeking access to the network is known as the Supplicant, a direct reference to the role its PAE adopts in EAP exchanges, similarly the system providing the point of access to the network is the Authenticator.

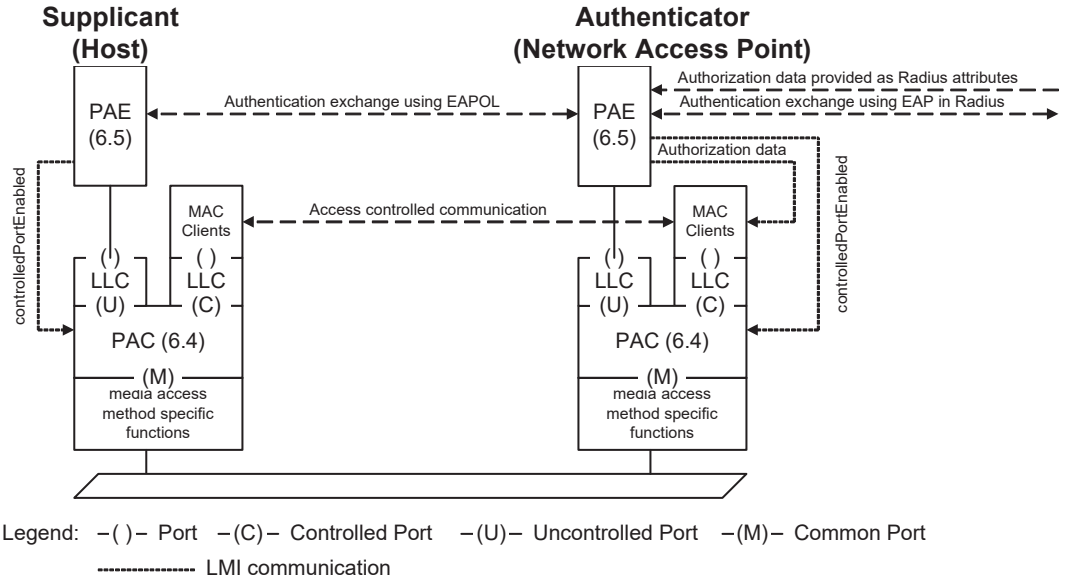


Figure 7-2—Network access control with a physically secure point-to-point LAN

The Authenticator's PAE can use EAPOL to announce the network access point's capabilities, and the Supplicant can use these announcements to influence its choice of EAP parameters and to attempt to access a particular network by choosing an advertised NID. If the Authenticator and Supplicant implement MKA and cache CAKs, roaming access to the network without prior reauthentication (6.1) can be supported, though the Authenticator can be required to discard any cache CAK if authorizations are no longer valid.

Where the Authenticator function is provided by a VLAN-aware Bridge, the relationship between the PAE, the PAC, and the Bridge's MAC Relay Entity is shown in Figure 7-3.

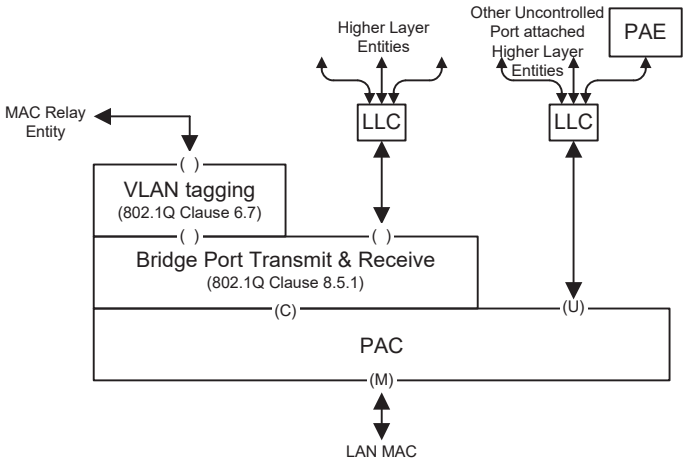


Figure 7-3—Network access controlled VLAN-aware Bridge Port with PAC

7.1.3 Connectivity to unauthenticated systems

A VLAN-aware Bridge that implements port-based network access control can use two general methods to provide connectivity to unauthenticated systems attached to a secured network. First, frames relayed to and from the unsecured port can be confined to those received from and transmitted on an 'Unauthenticated VLAN' within the secured network by appropriate configuration of the Port's PVID, Static VLAN Registration Entries, and Enable Ingress Filtering parameters.

NOTE 1—Direct communication between attached unauthorized systems can also be prevented by appropriate VLAN configuration. However discussion of all the potentially useful VLAN configurations is outside the scope of this standard. The present clause is limited to setting and explaining requirements for port-based network access control.

NOTE 2—VLANs are commonly configured to support IP subnets, thus traffic segregation can be maintained through a routed infrastructure. Frames transmitted by routers and end systems that do not have VLAN interfaces can be VLAN-tagged using port-based ingress rules or port-and-protocol classification. For further discussion of traffic segregation, see IEEE Std 802.1Q.

The second general method that bridges (and other intermediate systems) can use to provide controlled connectivity to unauthenticated systems is to restrict relayed frames to a small subset known not to pose any threat, thus preserving the principal port-based network access control objective—restricting network disruption and attacks in general caused by frames transmitted by unauthenticated systems to their individual LANs of origin. In practice (without the use of a VLAN to segregate such frames or allow their explicit identification by a secured source) the safe subset is small and limited to frames transmitted to, rather than received from, the unauthenticated systems and to frames safe to transmit without confidentiality protection or at least the level of confidentiality implied by not disclosing data at random. Implementations of this standard should limit such frames to those whose reception can be required before unauthenticated systems will devote resources to participate in authentication, key agreement, and secure data transfer protocols. The Wake-on-LAN protocol (Annex E), when identified by a specific UDP port, meets these criteria.

To prevent data loops, and carry out other maintenance and monitoring activities, a bridge port's higher layer entities need to operate even when data relay is expected to be limited to just a few frames. Therefore, the Controlled Port needs to be operational. The authentication and authorization, or rather the lack of it, associated with the Controlled Port is communicated to its clients, including the bridge's MAC Relay Entity, so they can implement appropriate filtering and policy controls.

NOTE 3—An example of such a policy is setting restrictedRoot and restrictedTcn for RSTP, so an unauthenticated system cannot become the root of the spanning tree or inject spurious topology changes into the rest of the network.

While the above description suggests that the bridge's MAC Relay Entity needs to implement sufficiently fine grained filtering, and possibly other frame inspection procedures, to ensure that only frames of the particular subset are forwarded, this is not the only possible model and not the one most readily adapted to most bridge implementations. An alternative is for the ingress and egress controls provide by the MAC Relay Entity to be set to filter all data, and for a selective relay entity within the bridge to receive the frame, inspect it to ensure that it belongs to the restricted subset, and to forward it (subject to the spanning tree Port States or other active topology constraints that the MAC Relay Entity would apply to forwarded data). Frames received from unauthenticated ports should not be relayed in this way.

To support this model, and be amenable to widespread deployment, the destination address of frames specifically designed to be relayed to unauthenticated systems should be the unicast address of the management or other secured port of the bridge or a group address, and the protocol should have a well defined Ethertype or subsequent protocol identifiers and should not have bulk data transfer as its objective. These requirements allow the selective relay entity to function as an end station connected to the network, as do the other entities responsible for operating bridge protocols (see 8.13 of IEEE Std 802.1Q-2018).

Figure 7-4 illustrates provision of both secure and controlled unsecured connectivity to the individual LAN attached to the Bridge Port on the right hand side of the figure.

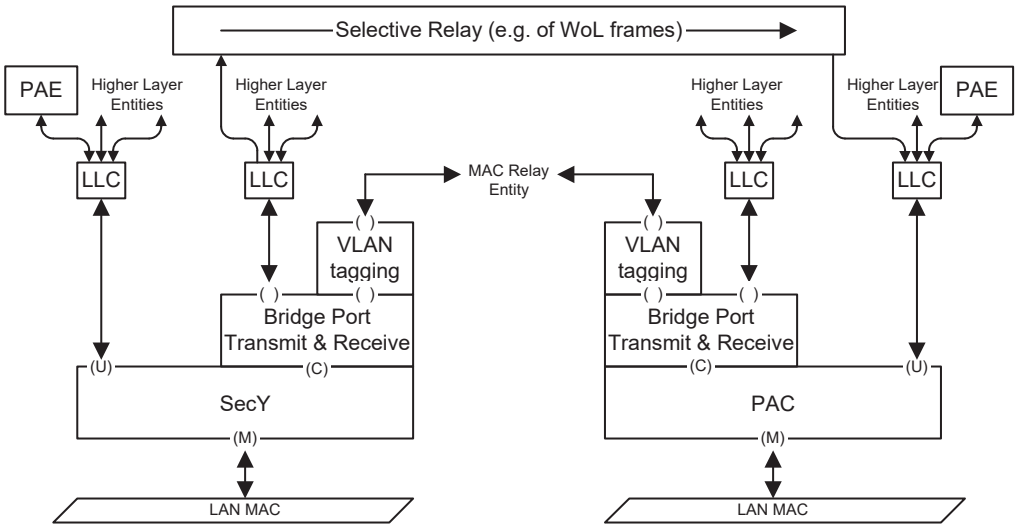


Figure 7-4—Selective relay to a physically secured unauthenticated port

7.2 Infrastructure support with physically secure LANs

In a number of important network infrastructure scenarios, the primary challenge is not the lack of positive intent on the part of those who have access to the equipment and physical media, but the potential for configuration and wiring errors. For example, fiber can pass through common patch panels that serve many network users and providers. Port-based network access control can be used to verify initial and continued correct connectivity, or to identify communicating systems and interfaces prior to invoking configuration parameters appropriate to that connectivity.

In other cases the cost of mounting an attack on the physical connectivity is both high and unlikely to benefit the attacker, since sensitive data will be subject to end-to-end encryption, but the cost of incorrectly identifying a connection is considerable, and appearing to offer an opening to an attacker is embarrassing even if security is not actually compromised. For example, port-based network access control can be used to identify connections between service providers and their customers.

These scenarios are illustrated in Figure 7-5.

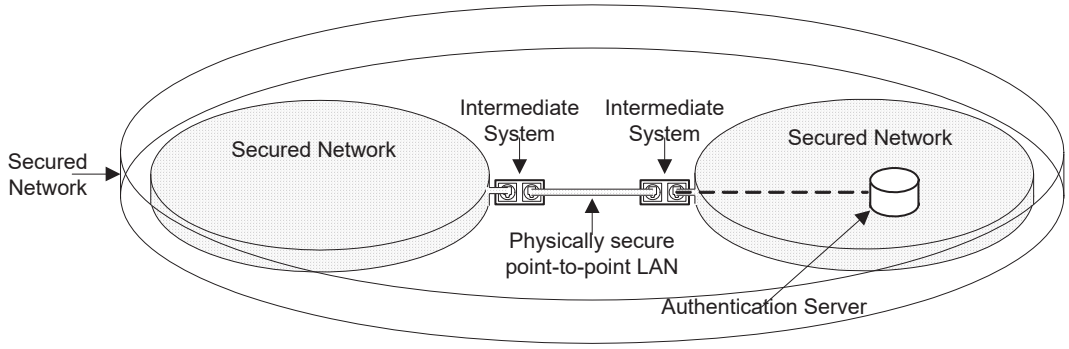


Figure 7-5—Network infrastructure with a physically secure point-to-point LAN

7.2.1 Assumptions and requirements

The port-based network access control application described in this subclause (7.2) is characterized by the following assumptions and requirements:

a) Security of individual links within the network is not of primary concern. Deployment of link authentication support on existing systems that lack the necessary hardware can be higher priority.

b) The communicating systems are similar, if not identical, and there is not necessarily any clear distinction between their roles within the network.

c) Network connectivity, through the communicating systems, is to be provided whenever possible. The systems can form part of the path from other systems to critical network resources, such as a AAA server. If access were to be dependent on the network as a whole being mostly operational a *chicken and egg* scenario might apply, with recovery from a power failure (for example) that affected a number of systems being impossible.

d) It should be possible to reinstate prior connectivity without requiring immediate access to centralized network resources, such as a AAA server.

e) The number of such links an intermediate systems in a network can be small, and local configuration of authentication can be desirable.

f) Interruptions in communication, e.g., while reauthentication occurs, are unacceptable. Infrastructure links have to be capable of providing years of uninterrupted service to meet reliability.

g) Handling error conditions by active human participation is either prohibitively expensive or completely impractical. A human interface is typically not available at the time of authentication, nor is there are an operator present.

h) The network configuration is expected to remain unchanged for long periods of time, with movements or changes in the communicating system or their connecting physical link being rare.

7.2.2 System configuration and operation

The interface stack appropriate to each port of a VLAN-aware bridge is as illustrated in Figure 7-3. The PAE may or may not implement PACP to support EAP exchanges, but does implement MKA so that authentication can be rapidly confirmed by an exchange local to the two intermediate systems. MKA also allows both systems to implement both EAP system roles (as in Figure 7-10) by providing the means for the elected MKA Key Server to select between the results of two separate authentications. The systems are moved or redeployed infrequently so announcement and selection are not required.

7.3 Host access with MACsec and point-to-point LANs

This application of port-based network access control is similar to that described in 7.1, but recognizes that there are many opportunities for an attacker to insert additional equipment into a supposedly secure physical connection without that being apparent to an accessing host user. MACsec is used to secure the LAN that provides that connection. See Figure 7-6.

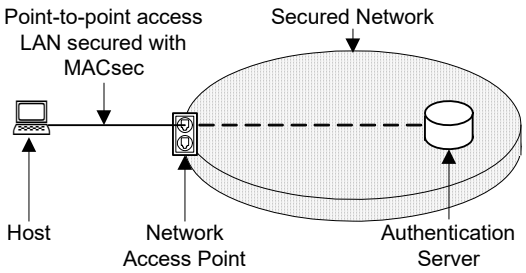


Figure 7-6—Network access control with MACsec and a point-to-point LAN

7.3.1 Assumptions and requirements

The assumptions and requirements of this application are generally those described in 7.1.1, with the exception of the need to secure the LAN providing the network access connection. The guarantees provided by MACsec in securing that connection are described in Clause 6 of IEEE Std 802.1AE-2018.

The use of cryptographic security changes the prior (7.1.1) assumptions and requirements as follows:

a) Cryptographic keys are generated during the authentication exchange and are used to generate session keys as described in 6.2.

b) While the point-to-point LAN is intended to provide connectivity to one other system, defending against an intruder attempting to deny service can require positive selection of the Network Access Point or participation in alternative authentication exchanges to the point of success.

Hosts can participate in authentication exchanges to acquire fresh master keys for cryptographic security. This scenario thus extends the applicability of port-based network access control to systems that are known to be permanently connected to the same network access point. Such systems can be mission critical, and their availability can be fundamental to the operation of the network itself. The requirements placed on this application scenario by such systems are similar to those for infrastructure connectivity (7.4), and differ markedly from those of hosts accessing charged public networks:

c) Communication cannot be interrupted by reauthentication, or associated changes in master keys, unless the systems are no longer authorized to access the network.

d) Connectivity to the network is to be restored following failure and restoration of the access LAN, even if connectivity to other network systems (including AAA servers providing authentication) is not yet available.

e) The need for active human participation, directly or through network management, is to be minimized and can be assumed to severely impact reported system availability.

7.3.2 System configuration and operation

The processes and entities that support this application are illustrated in Figure 7-7. The host that is seeking access to the network is known as the Supplicant, a direct reference to the role its PAE adopts in authentication exchanges, similarly the system providing the point of access to the network is the Authenticator.

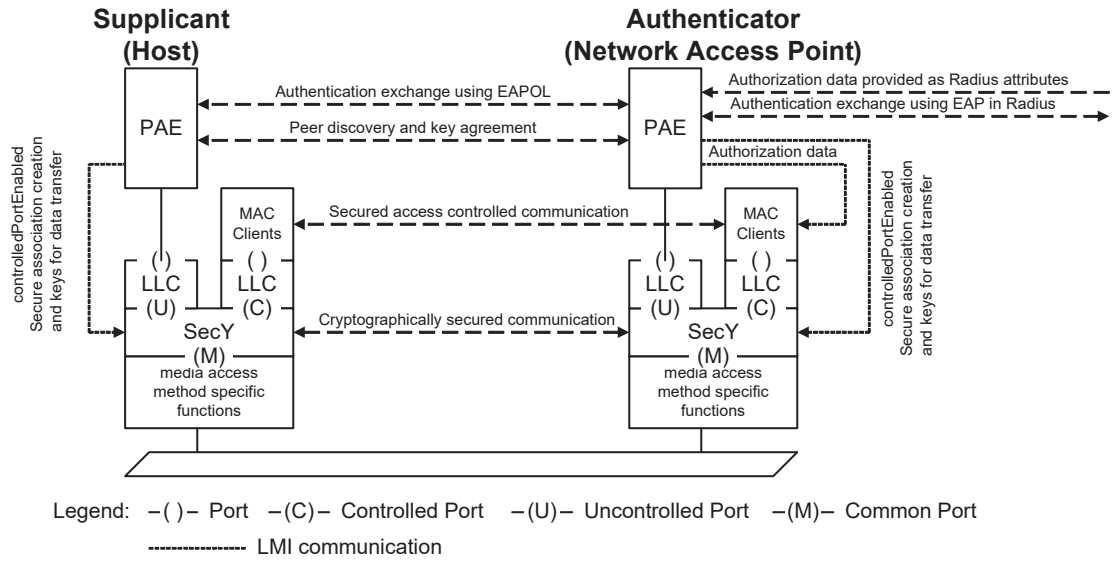


Figure 7-7—Network access control with MACsec and a point-to-point LAN

7.3.3 Connectivity to unauthenticated systems

A VLAN-aware Bridge that implements port-based network access control with MACsec can use the general methods to provide connectivity to unauthenticated systems introduced in 7.1.3. To maximize interoperability use while providing controlled connectivity to unauthenticated hosts use of the system configuration described in 7.5.3 and illustrated by Figure 7-13 is recommended.

7.4 Use with MACsec to support infrastructure LANs

Port-based network access control can be used to secure both point-to-point (Figure 7-8) and shared media connections (Figure 7-9) between the intermediate systems, such as bridges and routers, that compose a secure network.

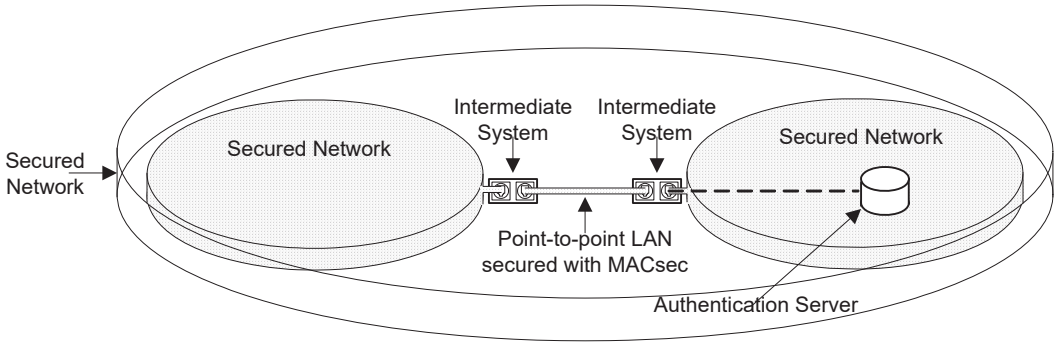


Figure 7-8—Point-to-point LAN within a secured network

In both figures, MACsec is shown as being used to connect two or more separate secure networks to form a larger secure network. Each of the separate networks can comprise further intermediate systems communicating securely using MACsec, or can be limited to the intermediate system itself. In this way, the secured network can be extended.

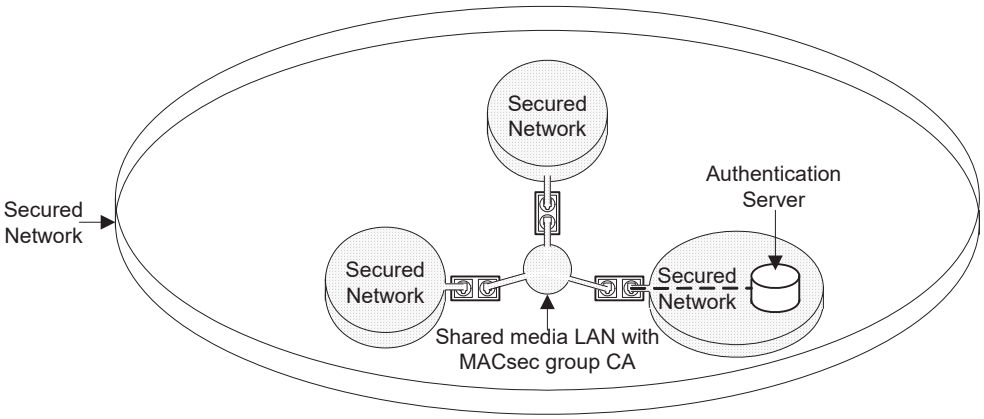


Figure 7-9—Shared media LAN within a secured network

While it is possible to support multiple CAs on any LAN, only one should be used where that LAN supports network infrastructure. This recommendation, also made in IEEE Std 802.1AE, ensures that the network topology does not change when security is enabled or disabled.

NOTE—Shared media connectivity in network infrastructures is typically provided by a network operator using a Provider Bridged Network (IEEE Std 802.1Q) comprising physical point-to-point LANs and bridges to offer virtual LAN services to a number of customers.

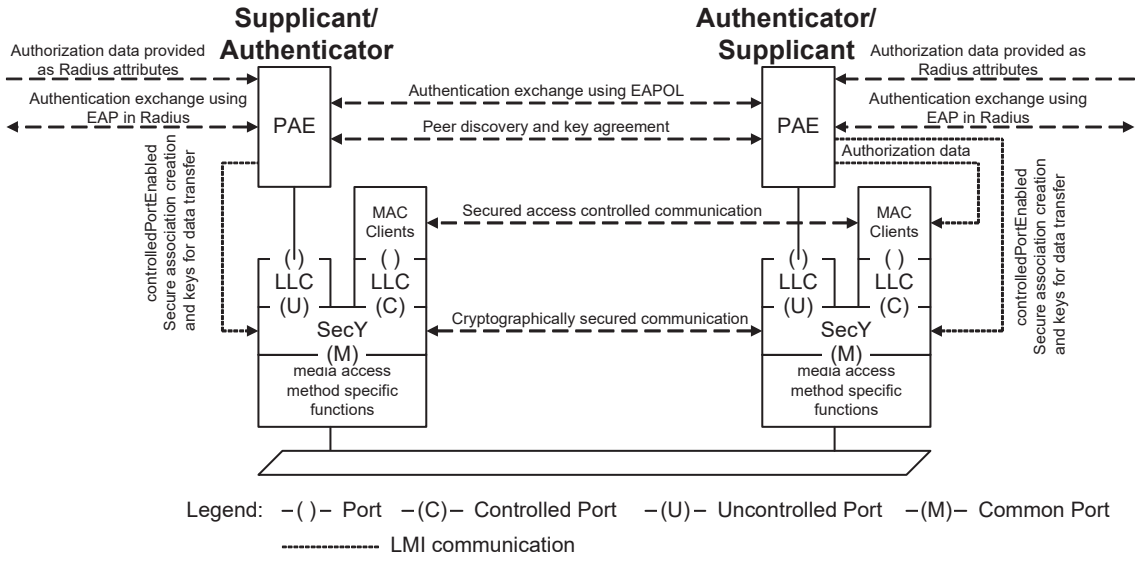


Figure 7-10—Network access control within the network infrastructure

7.4.1 Assumptions and requirements

The port-based network access control application described in this subclause (7.4) is characterized by the mission critical requirements introduced in 7.3.1 and the following:

a) In most cases the peer systems that wish to communicate have no way of determining which is to be the Supplicant and which the Authenticator.

7.4.2 System configuration and operation

The processes and entities that support this application are illustrated in Figure 7-10.

7.4.3 Connectivity to unauthenticated systems

Connectivity to unauthenticated systems within the infrastructure of a secured network is undesirable. Controls to aid staged deployment of MACsec in a network, together with management counters designed to verify the current state of deployment, are specified by IEEE Std 802.1AE (see 10.5 and 10.6 of IEEE Std 802.1AE-2018).

7.5 Host access with MACsec and a multi-access LAN

A shared media LAN, providing a multipoint connectivity association between stations connected to that LAN, can be used to provide the equivalent of individual point-to-point connections from one station that provides and controls access to a network, to each of the others. Data for each of the connections is secured and kept separate from the data for the others by using MACsec. See Figure 7-11.

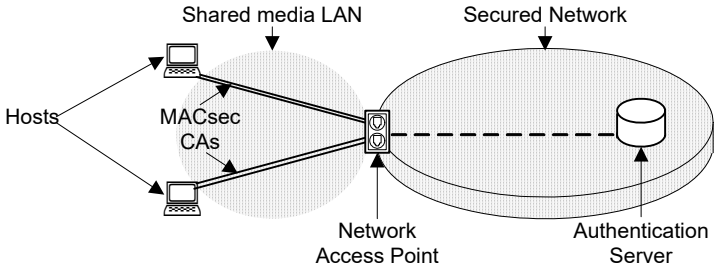


Figure 7-11—Network access control with MACsec and a multi-access LAN

Virtual ports, each including a SecY, are created as required by the network access point. Secure connectivity between the stations that connect to the network access point is provided only by the functionality within that access point, typically bridging or routing.

NOTE 1—This standard specifies the procedures necessary to create virtual ports, but the instantiation of such ports within a bridge or router, though easy to envisage, is outside its scope. Creation of such virtual ports within the architecture specified in IEEE Std 802.1Q would require support for transmission of frames received on a physical Bridge Port through that same port if any two virtual ports can be members of the same VLAN.

The individual MAC Addresses of the hosts can be used to distinguish between each of the potential two member CAs. EAPOL frames from each host to the network access point carry that address in their source address field, while frames from the network access point that are explicitly intended for a particular host carry that host's address in the destination address field.

NOTE 2—EUI-48 identifiers, including globally unique MAC Addresses, are intended to identify items of real physical equipment or parts of such equipment such as separable subsystems or individually addressable ports.[17]

NOTE 3—This standard makes use of a layered protocol model (see IEEE Std 802.1AC), thus allowing the entities and protocols it specifies to be instantiated above a service access point supported by an arbitrary interfaces stack (see Figure 7-18, for example). If a single physical realization of an IEEE 802 LAN MAC were to be identified by more than one MAC Address, a separate MSAP would be provided for each address and the resulting system behavior would, for the purposes of this standard, be the same as that of a collection of separate stations each with a single address.

[17] See the "Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)" provided by the IEEE Registration Authority.

7.5.1 Assumptions and requirements

The port-based network access control application described in this subclause (7.5) is characterized by the following assumptions and requirements:

a) A single LAN, and one physical point of attachment from the network access point to that LAN, are intended to support independently secured connections to each of the hosts.

b) The hosts are not required to communicate directly with each other securely, and indeed will not be capable of so doing. All such communication takes place through the network access point.

Assumptions and requirements common to this and prior host access applications include the following:

c) There is a clear physical distinction between the network equipment supporting access, on the one hand, and the hosts accessing the network on the other.

Both mission critical and typical single-user hosts can attach to the network using a multi-access LAN. It is therefore possible that both the typical single-user access requirements, for access only when the network is fully operational and can record or charge for access (as described in 7.3.1), and mission critical access requirements for continuous connectivity (as described in 7.5.1 above) apply to different hosts with connectivity through the same network access point.

7.5.2 System configuration and operation

The processes and entities that support this application are illustrated in Figure 7-12.

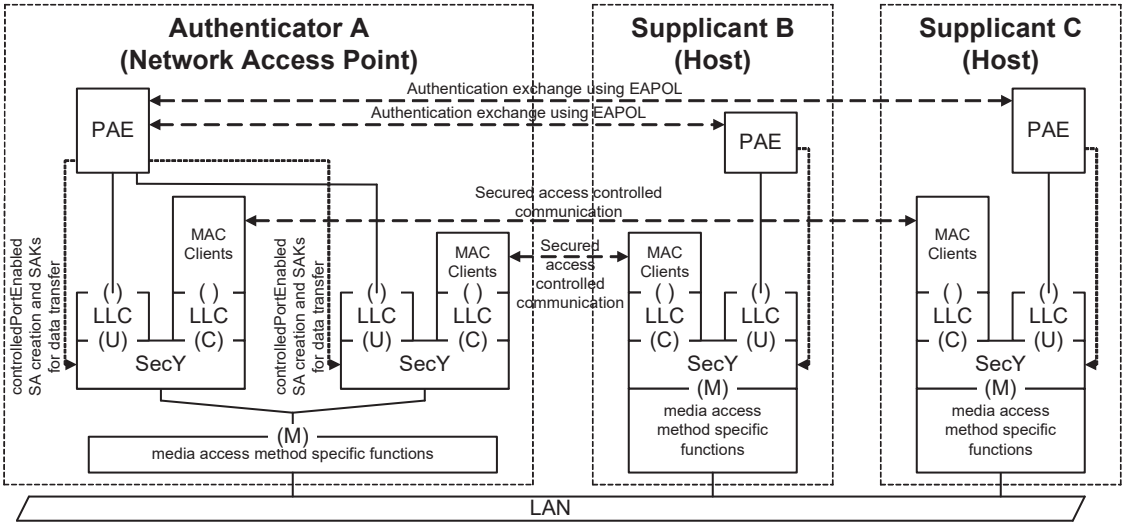


Figure 7-12—Network access control with MACsec and a multi-access LAN

7.5.3 Connectivity to unauthenticated systems

A VLAN-aware Bridge that implements port-based network access control with MACsec can use the general methods to provide connectivity to unauthenticated systems introduced above (7.1.3). It cannot be assumed that all unauthenticated systems are capable of decoding frames that include a SecTAG, even if the frame is not confidentiality protected. The systems that possess the necessary keys to validate a given SecTagged frame compose one connectivity association, while those that can receive unprotected frames on the same individual LAN compose another, the first being a subset of the second. In contrast to the use of a PAC, MACsec allows any receiver to clearly distinguish between frames transmitted through a Controlled Port (with a SecTAG) and those transmitted through an Uncontrolled Port, so the connectivity association that provides unauthenticated connectivity can be associated with a separate port on a VLAN-aware Bridge that provides the network access point functionality illustrated in Figure 7-12. Figure 7-13 illustrates the system configuration of the Bridge Ports that are attached to a single multi-access LAN.

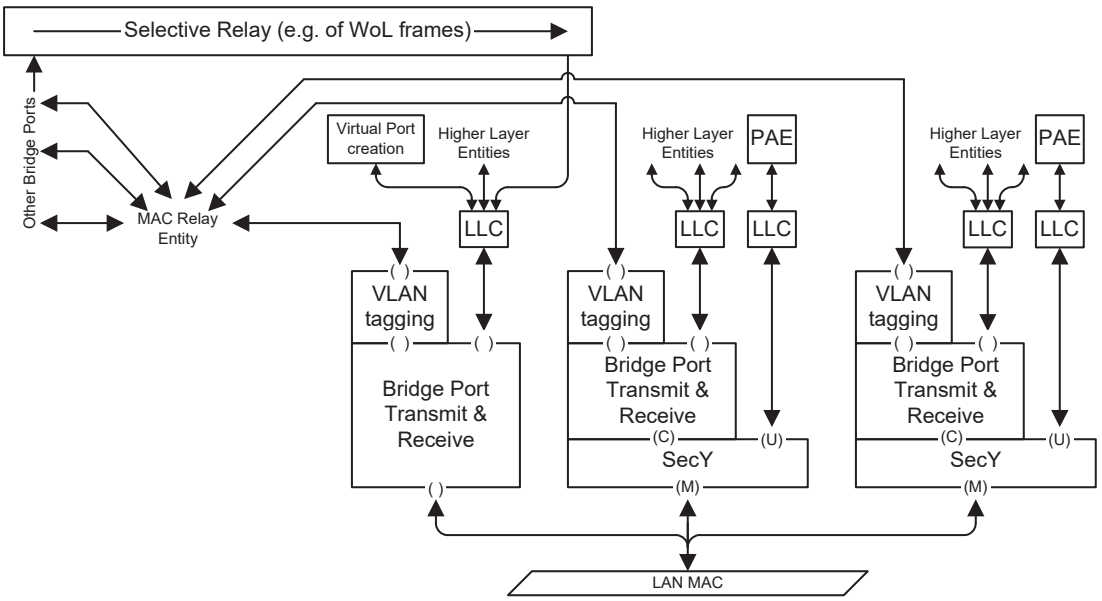


Figure 7-13—Secure and unsecured connectivity on a multi-access LAN

One Bridge Port (to the left in the figure) provides unsecured connectivity to the multi-access LAN, either by supporting selective relay of frames or by providing ingress and egress to one or more *Unauthenticated VLANs*. Port-and-protocol VLAN classification is configured for this port to prevent frames with the MACsec or EAPOL Ethertypes from being relayed. EAPOL frames with group destination MAC addresses will be forwarded or filtered by the bridge as appropriate to the scope of each address (11.1.1).

Each of the other ports provide communication to a single host attached to the LAN, frames from that host being distinguished by the SCI that is part of the SecTAG in each frame. These ports are *virtual*, that is they are created on demand by the PAE as described in 12.7.

NOTE 1—IEEE Std 802.1Q-2018 does not specify a way to create a template for virtual ports, to allow the control over values assigned to managed objects for the port when it is created. Specification of such a template is for future study.

Use of the SCI allows separate secure connections to be maintained from a system with a single MAC Address. One or more of those connections could be to bridges that relay frames to further LANs, though use of a multi-access LAN within the core of a secured network is not recommended as the network topology would then depend on the use of MACsec to separate traffic, and thus imposes constraints on the methods use to stage deployment and diagnose system configuration errors.

The bridge's Spanning Tree Protocol Entity is attached to each of the ports (as specified by IEEE Std 802.1Q) including the virtual ports, thus preventing data loops. Frames from unauthenticated VLANs that are transmitted by the MAC Relay Entity port providing unsecured connectivity can also be transmitted through the secured virtual ports, if permitted by the network administrator. Attached hosts need to be prepared to receive them twice if configured to receive both MACsec protected and unprotected frames.

NOTE 2—Two different network/system configurations are possible for frames assigned to unauthenticated VLANs. In one, the same VLAN is used both for communication from unauthenticated systems and for communication to those systems. If egress through a secured virtual port is permitted for that VLAN, an attached host needs to interpret each frames' VLAN tag to distinguish those from secured sources. This configuration requires accurate configuration of potentially large numbers of systems and is thus error prone. The other configuration uses one VLAN to carry frames from unauthenticated systems to chosen servers within the secured network, and another for frames whose delivery to unauthenticated systems is permitted even though they been transmitted by authenticated systems. Such VLAN pairs can be configured to support a single IP subnet. For further details, see IEEE Std 802.1Q.

7.6 Group host access with MACsec

While a multi-access LAN (7.5) provides independent and separate access for a number of hosts through a single network access point, there are application scenarios where direct communication between the hosts is also desirable. Access to a network is often enforced in a wiring closet per desktop LAN, while there can be two or more LAN stations per desk. The combination of a PC and an IP phone, interconnected by repeater-like functionality is typical. Communication between the PC and phone for computer assisted telephony, for example, is direct and does not pass through the network access point, while data from both PC and phone goes directly to the network. See Figure 7-14.

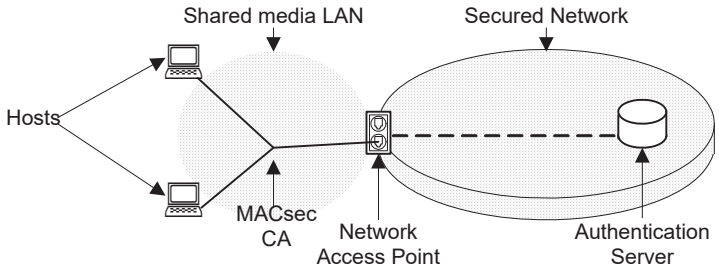


Figure 7-14—Group host access

The use of MACsec to support a group CA in this scenario secures all the data communication described and does not require the instantiation of multiple virtual ports per physical port at the network access point, or bridging between those ports at the access point or within the secured network. Pairwise mutual authentication takes place between the network access point (acting as an EAP Authenticator) and each host (acting as an EAP Supplicant) prior to the network access point distributing the CAK for the group CA to the host. The network access point dynamically creates PAE instances to support each pairwise authentication as required.

It is possible to combine the use of group host access with point-to-point access over the same individual shared media LAN, first authenticating each host and then allocating it to an appropriate group on the basis of that authentication. Such a combined scenario allows, for example, a group of systems under the control of a single user to communicate directly, while requiring communication with another group to occur through the network access point.

NOTE—The combined scenario described immediately above requires each group to be on a separate VLAN, with independent source address location learning between those VLANs, and connection between the VLANs being provided by routing, either within the network access point or elsewhere within the secured network.

7.6.1 Assumptions and requirements

The port-based network access control application described in this subclause (7.6) is characterized by the following assumptions and requirements:

a) A number of stations connected to an individual shared media LAN wish to communicate directly both with each other, and with other stations that are accessed through a network access point.

b) Permission to join the group, given by supplying the necessary CAK, is under control of the network access point that mutually authenticates each accessing system directly and makes use of a AAA infrastructure.

c) The network access point is always present and acts as the MKA Key Server.

7.6.2 System configuration and operation

The processes and entities that support this application are as illustrated in Figure 7-12, and a Group CAK is distributed as described in 6.2 and Figure 6-4. However once the group CAK is distributed all the stations attached to the shared media LAN exchange MKPDUs directly, and communication between attached stations can continue even if the Network Access Point is powered down or otherwise unavailable.

7.7 Use with MACsec to support virtual shared media infrastructure LANs

While the use of physical shared media has declined over the years, multipoint-to-multipoint services can be provided by S-VLAN bridges operating transparently to C-VLAN bridges as specified by IEEE Std 802.1Q for Provider Bridged Network and Provider Backbone Bridged Networks. MACsec can be used to secure communication, across and or to that PBN or PBBN, for the different geographically dispersed sites of a single enterprise, as illustrated in Figure 7-15.

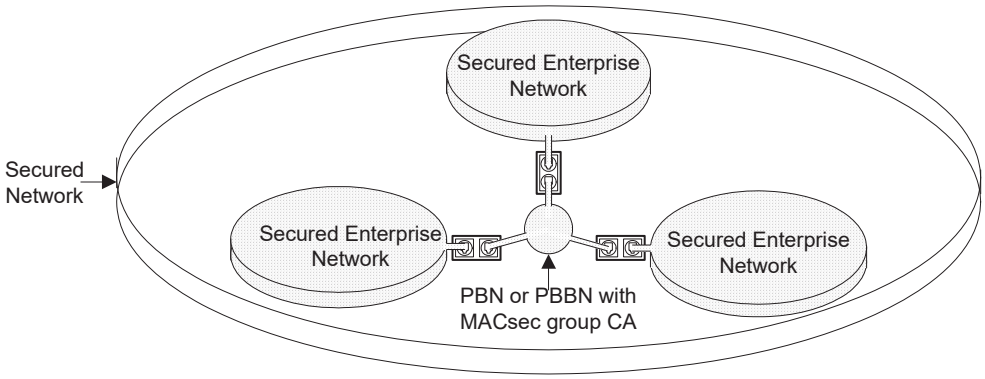


Figure 7-15—Multipoint connectivity across a Provider Bridged Network

7.7.1 Assumptions and requirements

The port-based network access control application described in this subclause (7.7) is characterized by the following assumptions and requirements:

a) The PBN is under the administrative control of a different organization from that responsible for the individually secured enterprise networks, transports data for many different enterprises including direct competitors, relies on provisioning and configuration systems to ensure that data is not accidentally shared, uses transmission and switching facilities that are not guaranteed secure to a level that can be required by a given enterprise, and charges for service.

b) The enterprise may wish to secure connectivity across the PBN, independently of the operator of the PBN and the systems within it.

c) The enterprise or the operator of the PBN may wish to secure connectivity to the PBN, to prevent theft of service or compromising data as it is transmitted from the PBN to the enterprise as that can be the greatest point of vulnerability.

d) An enterprise's connectivity across the PBN should be maximized at all times, and not dependent on the systems at any one site.

e) Within the enterprise the responsibility for maintaining infrastructure connectivity across the PBN is likely to be somewhat separate from that responsible for host connectivity. The organization responsible is likely to use different infrastructure support systems, and can be separate from that responsible for maintaining connectivity at each site.

f) Handling error conditions by active human participation is either prohibitively expensive or completely impractical. A human interface is typically not available at the time of authentication, nor is there are an operator present.

7.7.2 System configuration and operation

Figure 7-16 illustrates the internal organization of the MAC sublayer in a Provider Bridged Network, highlighting communication between

a) An enterprise's Customer Bridge or other equipment, across the PBBN.

b) Adjacent S-VLAN aware Bridges, within the PBBN.

c) An enterprise's Customer Bridge and the PBBN.

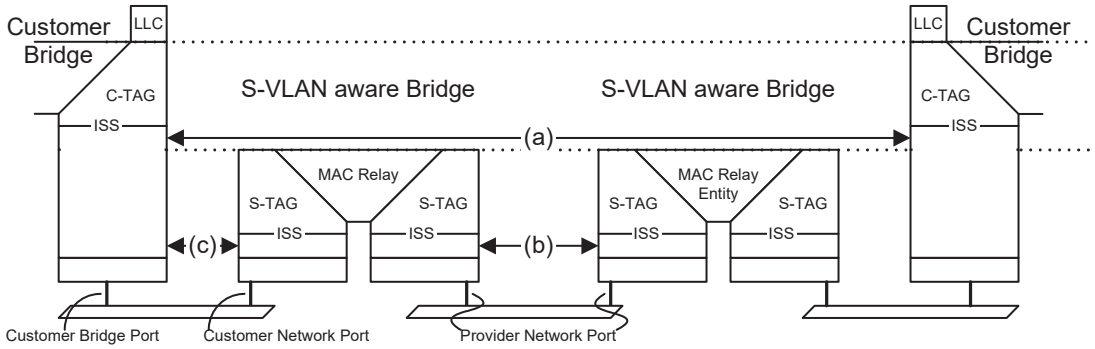


Figure 7-16—Internal organization of the MAC sublayer in a Provider Bridged Network

NOTE—Figure 7-16 is based on Figure 15-1 of IEEE Std 802.1Q-2018.

If it is the intention to secure communication for both a) and c), then the use of two independent SecY's within the Customer Bridge Port's interface stack is required as shown in Figure 7-17. The interface stack shown includes the service access priority selection function described in 6.13 of IEEE Std 802.1Q-2018.

If communication across the PBN is to be secured, but communication between the enterprise's Customer Bridge and the Customer Network Port in the PBN is to be treated as physically secure but authenticated to guard against accidental miswiring, then the interface stack shown in Figure 7-18 can be used. The interface stack of Figure 7-17 can also be used for this application, provided that Logon Process controls (12.5) for the lower PAE permit unsecured connectivity. Similarly both interface stacks can be allowed to interoperate with systems that lack port-based network access control capabilities.

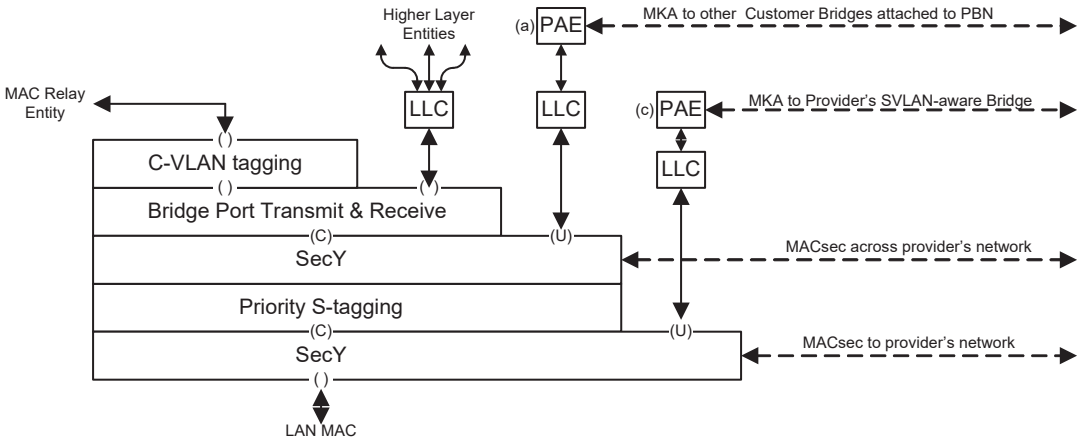


Figure 7-17—Secure PBN transit and access with priority selection

Both Figure 7-17 and Figure 7-18 emphasize the PAE's use of MKA, as it is likely that few bridges (or routers) will be connected to a single PBN multipoint service for any given enterprise network (at the time this standard was developed the expected number is four or five, with more than twenty being rare) and will typically be configured with a pre-shared Group CAK to avoid depending on potentially unreachable Authentication Servers. If a Group CAK is to be distributed by a Key Server, following the use of EAP to distribute a pairwise CAK for the Key Server and each potential member of the CA, then the Group CAK should be cached to allow rapid resumption of secure connectivity following power failures and other interruptions.

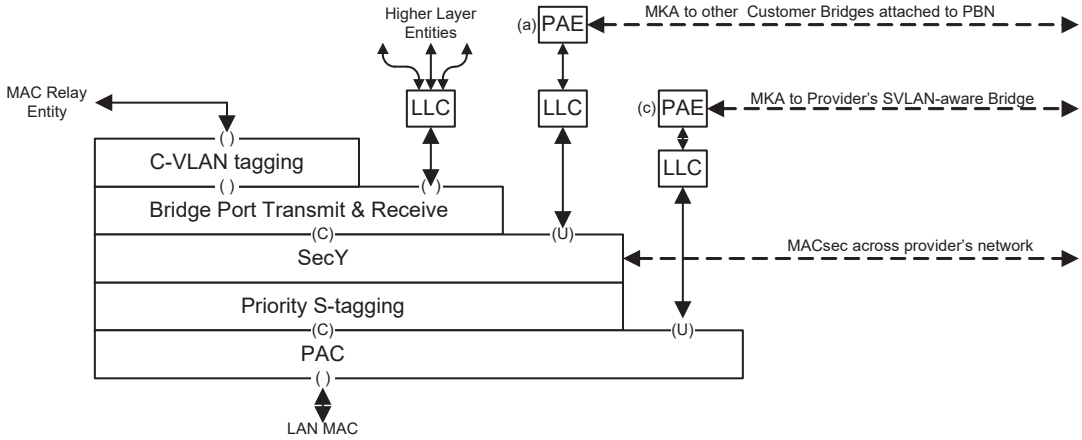


Figure 7-18—Secure PBN transit and with priority selection

8. Authentication using EAP

The Extensible Authentication Protocol (EAP, specified in IETF RFC 3748 [B14]) can be used to mutually authenticate a Supplicant PAE (Port Access Entity, see 6.3) and an Authenticator PAE, each associated with a Port attached to the same LAN. EAP is a general protocol that supports multiple authentication mechanisms including the use of Kerberos, Public Key Encryption, and One Time Passwords.

Each PAE (6.3, Clause 12) may implement the Port Access Control Protocol (PACP) and a higher layer supported by PACP. The Supplicant's higher layer provides EAP functionality, while the Authenticator's higher layer combines EAP and authentication, authorization, and accounting (AAA) functionality. In both PAEs, the higher layer reports the result of an authentication attempt as a Success, Fail, or Timeout, and successful authentication can be accompanied by the secure delivery, to both PAEs, of a secret key that can be used to prove mutual authentication and to distribute or agree further secret keys (6.2). Each PAE may also be configured to use a pre-shared key (PSK) instead of EAP, or as a fall-back if EAP authentication attempts fail. In these cases, selective distribution of the PSK constitutes the authentication.

PACP initiates authentication attempts by the PAE higher layer, retries initial authentication (if necessary) to guard against frame loss when the two communicating Ports are enabled at different times, provides periodic reauthentication, and terminates authentication on request. PACP uses EAPOL (EAP over LAN) PDU formats defined in Clause 11 of this standard. These formats are used for all communication between PAEs. They support direct communication between the PACP entities, as well as an encapsulation format that allows EAP Messages, transmitted and received by the PAE higher layers, to be carried directly by the LAN MAC service. While PACP initiates and retries (and can abort) authentication attempts by the higher layers, all retransmission within an individual authentication attempt is carried out by those higher layer functions.

This clause specifies PACP, and the interface between PACP and the higher layer EAP and AAA functionality, in terms of state machines, state variables, and procedures. This model of operation is simply a description of the necessary functionality, and does not constrain real implementations: these can adopt any internal model of operation compatible with the externally visible behavior specified by this standard. Conformance to this standard is purely in respect of observable protocol. The notational conventions used in the state machines are described in Annex C.

Specification of the higher layer PAE functions is outside the scope of this standard, though this standard does require the use of an EAP method that provides mutual authentication when EAP is supported, places further constraints on the methods to be used in conjunction with MKA, and mandates the use of EAP-TLS (IETF RFC 5216) for integration with IEEE Std 802.1AR (8.11). EAP protocol exchanges are defined by IETF EAP standards, IETF RFC 3748 [B14], and successor standards. One example of a AAA protocol, RADIUS, and its use for "pass-through" forwarding of EAP Messages to an Authentication Server, is defined by the IETF RADIUS standards, IETF RFC 2865 [B6], IETF RFC 2866 [B7], IETF RFC 3579 [B12], and successor standards.

NOTE—IEEE Std 802.1X-2004 and prior revisions of this standard included a 'Backend Authentication' state machine that tracked the behavior of EAP. That machine has been removed, allowing for a wider range of EAP method behaviors, including re-authentication through the use of EAPOL-EAP frames without changing PACP state machine states (see IETF RFC 6696 [B26]).

Each EAP authentication exchange occurs between one Supplicant and one Authenticator. A PAE may instantiate multiple Authenticators, each participating in an exchange with a single Supplicant, to support a number of access controlled virtual ports (6.3.6) or to secure group communication (6.2, Figure 6-4). Each Authenticator is created and addressed following the rules for EAPOL frame transmission and reception (12.7, 11.1). The specification in this clause (Clause 8) is followed by each Authenticator independently of the existence of any others for the same Common Port.

8.1 PACP Overview

Figure 8-1 shows the exchange of EAPOL frames between Supplicant and Authenticator PAEs as a consequence of the operation of PACP and the EAP and AAA higher layers, and the interfaces between PACP and the higher layer in each PAE, and between PACP and the PAE's Logon Process (12.5). The latter controls the PAE's use of EAP and the potential use of PSKs or unsecured connectivity.

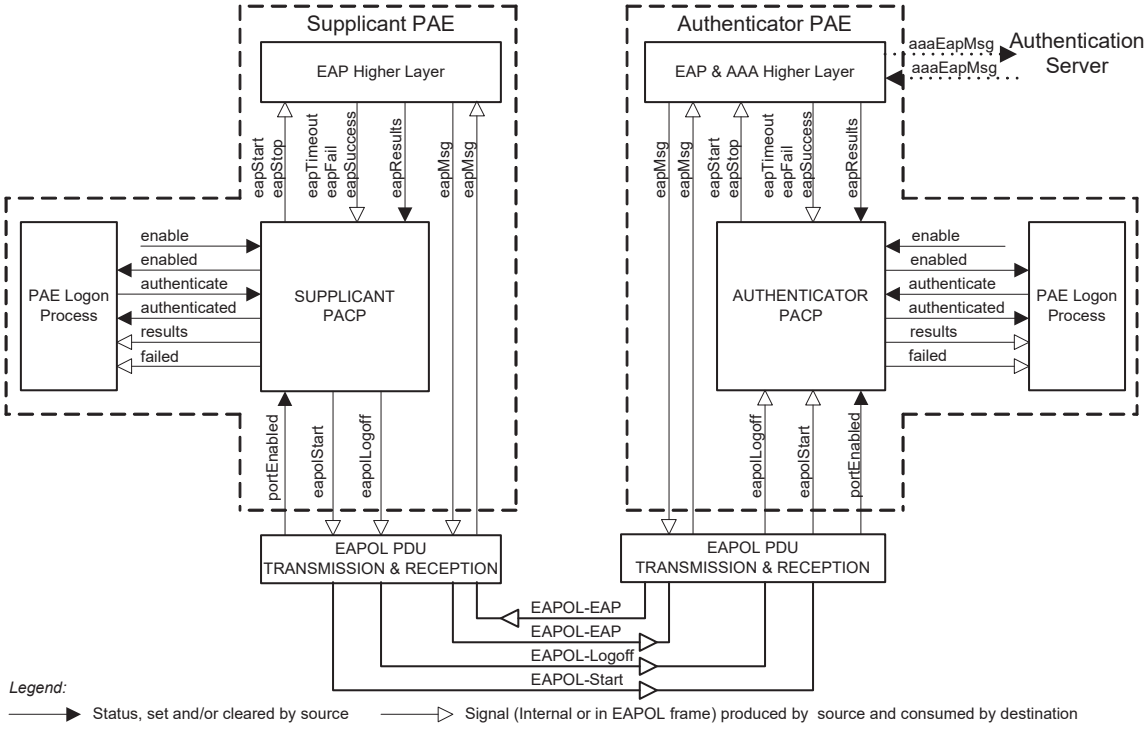


Figure 8-1—PAEs, PACP, EAP Messages, and EAPOL PDUs

Authentication can be initiated by the Supplicant PAE, by the Authenticator PAE, or by both PAEs, each at the request of its client (the entity that provides network access control for the associated Port), possibly as a result of the Port's MAC Operational state becoming true. EAP is a Request-Response protocol, with the Authenticator higher layer functions sending Requests and the Supplicant's providing the Responses. When an Authenticator PAE is asked to authenticate, its PACP entity tells EAP to start, and the first EAP Request message is transmitted. That can happen at a time when the Supplicant PAE is not enabled—it might be part of a system that is powered off, a human user who has to supply authentication credentials might not be logged on, or it might see the Port's MAC Operational state transition true slightly later—so when the Supplicant PAE is asked to authenticate it transmits an EAPOL-Start frame as well as telling its EAP higher layer to start. On receiving the EAPOL-Start, the Authenticator PACP tells its EAP higher layer to stop any authentication attempt that is in progress and to start once more with its first Request.

Once mutual authentication has succeeded, the Authenticator PACP can ask its EAP higher layer to start periodically, reauthenticating the Supplicant. The Supplicant remains authenticated while reauthentication attempts take place. Reauthentication can be enabled or disabled by management, and the reauthentication period modified.

NOTE 1—In the absence of direction provided by the AAA server the default settings for host application scenarios (7.1, 7.3, 7.5) are for reauthentication to be disabled, with a reauthentication period (when enabled) of 3600 s (one hour).

If the Supplicant's client no longer wishes to be authenticated, the Supplicant's PACP entity can transmit an EAPOL-Logoff that the Authenticator's PACP entity can use to terminate the EAP authentication. A Supplicant PAE that is using a PAC (6.4), and is not using MKA or IEEE 802.11 procedures to secure communication, should transmit an EAPOL-Logoff whenever the authentication credentials are user-based, and the user of the Supplicant system has logged off (in the case of an end system), or whenever the system has been reconfigured in a manner that would invalidate any previous authentication results (for example, a management change that affects the system's identity, or authorization to use the services of the Authenticator's system). When communication is secured by MACsec or IEEE 802.11 specific procedures (6.6), or supervised by MKA terminating an EAP authentication does not automatically terminate connectivity. In the case of MKA and MACsec, the PAE's Logon Process should continue to instruct the CP state machine to provide connectivity (see 12.3) until MKA fails. This requirement protects against the use of EAPOL-Logoff or EAPOL-Start frames in DoS attacks, once an EAP authentication has succeeded. To terminate connectivity either PAE needs to terminate MKA (and MACsec) operation. If an authentication will require the use of MACsec, MKA, or IEEE 802.11 procedures (if and when the authentication succeeds) an Authenticator PAE can discard any EAPOL-Logoff received. If a Supplicant or an Authenticator PAE needs to ensure that the results of an authentication cannot be used again, it deletes all state associated with the authentication, including MKA and EAP derived state (MSK and EMSK).

While EAP is a request-response protocol, and the EAPOL-Start frame capability is provided in recognition of that fact, neither the Authenticator nor the Supplicant PACP entities interpret the type of any particular EAP Message, be that a Request, Response, Success, or Fail, and do not constrain the way that EAP and the other higher layer functions determine success or failure, or police the request-response paradigm. If both a Supplicant and an Authenticator PAE are enabled for a given Port, and receive (as is likely) EAPOL PDUs for the same addresses, the EAP Messages received in EAPOL-EAP PDUs are delivered to both Supplicant and Authenticator higher layers. The EAP standards specify whether a Supplicant or Authenticator PAE processes or ignores any given EAP message type.

NOTE 2—Correct protocol operation depends upon the use of timer values by the Supplicant higher layer functions that are compatible with those used by the Authenticator's higher layer functions to retransmit EAP-Requests. There is no automatic means of communicating changes in timer values between Authenticator and Supplicant, so deviation from the default timer values can adversely affect the operation of the protocol.

8.2 Example EAP exchanges

Figure 8-2 uses the EAP-TLS mutual authentication method using ECDH/ECDSA (also applicable to RSA-DHE) to illustrate the operation of PACP and the encapsulation of EAP Messages in EAPOL frames, although the detailed content of the EAP-Request and EAP-Response exchanges is not fundamental to understanding the relationship of EAP and PACP. The figure shows "pass-through"—the forwarding of EAP messages to and from a separate Authentication Server using a AAA protocol (e.g., RADIUS, as described in IETF RFC 3579 [B12]) as a transport—as this accords with realistic implementation scenarios and best illustrates the relationship of EAP messages to EAPOL frames. Each EAPOL frame transmission is represented by an unbroken line while EAP messages carried in AAA protocol are shown by broken lines.

In an Authenticator-initiated exchange (as shown in Figure 8-2) all the EAPOL frames are of type EAPOL-EAP, i.e., they encapsulate EAP messages, and PACP is not aware that they convey EAP Request, Response, Success or Fail messages. PACP's role is limited to starting each PAE's higher layer as the port is enabled, and receiving the indication of success from the higher layer. In terms of the higher layer interface provided by PACP to EAP (Figure 8-1, 8.3), each EAPOL-EAP transmission and reception corresponds to an eapTxMsg or eapRxMsg respectively. The higher layer functions of the Authenticator PAE are responsible for forwarding the encapsulated EAP messages to and from the Authentication Server.

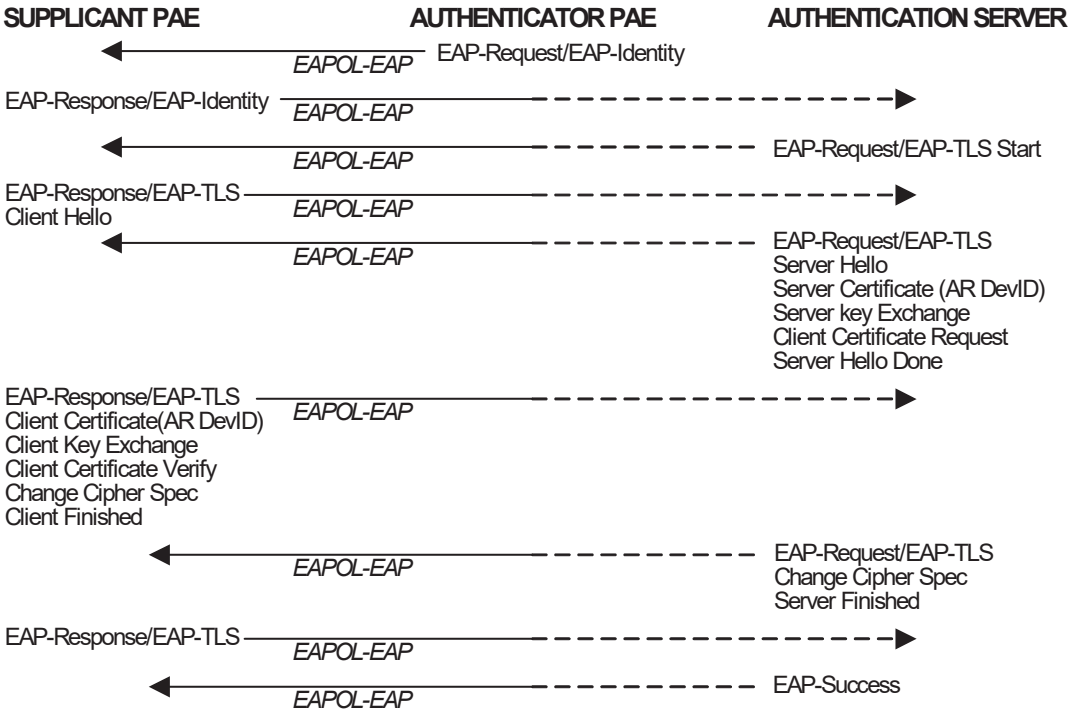


Figure 8-2—Authenticator-initiated EAP-TLS (success)

A Supplicant-initiated authentication conversation begins with an EAPOL-Start frame (see Figure 8-3) before proceeding as illustrated in Figure 8-2.

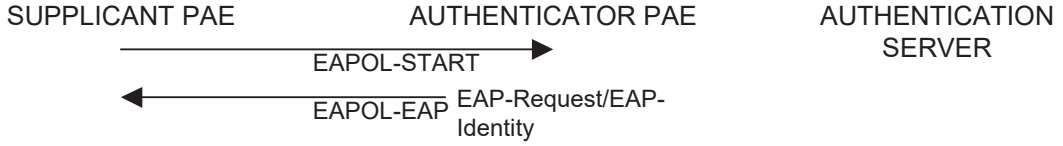


Figure 8-3—Supplicant-initiated EAP exchange

If the Supplicant does not support PACP and EAP authentication, but EAP authentication is enabled on the Authenticator, the Authenticator can retry transmission of its initial EAP-Request/EAP-Identity messages several times before abandoning its attempt to use EAP for mutual authentication. Similarly if the Supplicant attempts to use EAP authentication but the potential Authenticator cannot, the Supplicant can retry transmission of its initial EAPOL-Start.

8.3 PAE higher layer interface

The following variables and procedures compose the interface to the higher layer functions of each PAE:

— eapStop: Set by PACP to stop an ongoing authentication, if there is one in progress, and/or to initialize the higher layer. Cleared by the higher layer when initialized. After eapStop has been cleared the higher layer will discard all received EAP messages, and will not send a message, until eapStart is set.

  It is undefined whether one of eapSuccess, eapTimeout, or eapFail will be set after eapStop is asserted, but none of them will transition TRUE after eapStop is cleared until eapStart is set.

— eapStart: Set by PACP to start an authentication attempt, cleared by the higher layer to indicate that it is beginning that attempt, and will subsequently will return one (and only one) of eapSuccess, eapTimeout, or eapFail, unless preempted by an eapStop.

  Not set by PACP unless any prior eapStop has completed, and any prior eapStart has returned or has followed by eapStop. Not followed by an eapStart until clear.

  NOTE 1—It is not necessary for PACP to run a supervisory timer to ensure that eapTimeout is not delayed. The time elapsing prior to any timeout will depend on the higher layer's knowledge of its likely progress. The authenticator should return eapTimeout almost immediately if it cannot communicate with the Authentication Server (because it has no IP address yet, for example). However, the timeout is not so rapid that PACP has to delay before retrying eapStart, to avoid live-lock.

— eapTimeout: Set by the higher layer if the authentication attempt started by eapStart times out. Cleared by PACP prior to setting eapStart.

  NOTE 2—The higher layer is responsible for re-transmission within a single authentication attempt, and should protect communication with the Authentication Server with retransmissions appropriate to the transport use. PACP is not aware of that transport.

— eapFail: Set by the higher layer if the authentication attempt started by eapStart fails. Cleared by PACP prior to setting eapStart.

— eapSuccess: Set by the higher layer if the authentication attempt started by eapStart succeeds. Cleared by PACP prior to setting eapStart.

— eapResults: If eapSuccess is set, contains the results of the successful authentication.

— eapRxMsg: Set by the Port transmit and receive process to indicate receipt of an EAPOL-EAP message, cleared by the higher layer to indicate its receipt of the message.

— eapRxData: The received EAP message.

— eapTxMsg(eapTxData): A procedure provided by the transmit and receive process, and called by the EAP higher layer to transmit an EAPOL encapsulated EAP message.

The interface variables eapTimeout, eapFail, and eapSuccess are set by the PAE's EAP module to reflect the state of EAP method completion and do not necessarily correspond directly to the reception of EAP failure or EAP success messages. The PACP state machines require only that the EAP module indicate one of these results, or continue (an) authentication attempt(s) after eapStart has been set, unless eapStop is set, The operation of EAP module can be informed by parameters provided through its LMI, possibly by the Logon Process, and can make several authentication attempts before returning eapTimeout or eapFail. A Supplicant's EAP module might support a quiescent listening mode, waiting indefinitely for Authenticator initiation, and only set eapFail if such an authentication attempt fails. However either eapTimeout or eapFail has to be set if there is no longer any possibility of the authentication succeeding without transmission of a further EAPOL-START or action by the Logon Process. The EAP module's decision to set eapTimeout or eapFail can be based on parameters provided through its LMI at any time during its operation. After eapFail the Logon Process will have to clear supp.failed (see 8.4, Figure 8-6) if it wants to continue to attempt authentication (possibly after changing parameters supplied to the EAP module).

8.4 PAE Client interface

The following variables and procedures compose the interface provided by the PAE's Supplicant PACP entity and Authenticator PACP entity to the PAE's Logon Process. The latter uses the authentication result(s) (or failure of authentication) to make network access control decisions for the associated port:

— enabled: Set by PACP if the PAE can provide authentication.

  NOTE 1—enabled will be FALSE if the Port is not enabled, if the functionality provided by the PAE is not available, or not implemented, or the control variable enable has been cleared by management, e.g., because the application scenario authenticates a user and there is no user logged on.

— authenticate: Set by the PAE client to request authentication, and allows reauthentication while set. Cleared by the client to revoke authentication.

  To enable authentication the client also needs to clear failed (if set).

— authenticated: Set by PACP if the PAE is currently authenticated, and cleared if the authentication fails or is revoked.

— results: The results of the last successful authentication.

— failed: Set by PACP if the authentication has failed or has been terminated. The cause could be a Fail returned by EAP, either immediately or following a reauthentication, an excessive number of attempts to authenticate (either immediately or upon reauthentication), or the client deasserting authenticate. The PACP will clear authenticated as well as setting failed.

  NOTE 2— Any ongoing authentication exchange will be terminated (by the state machines) if enable becomes FALSE and enabled will be cleared, but failed will not be set.

The enabled, authenticate, authenticated, and failed variables may be read by management to monitor the current state and progress of authentication.

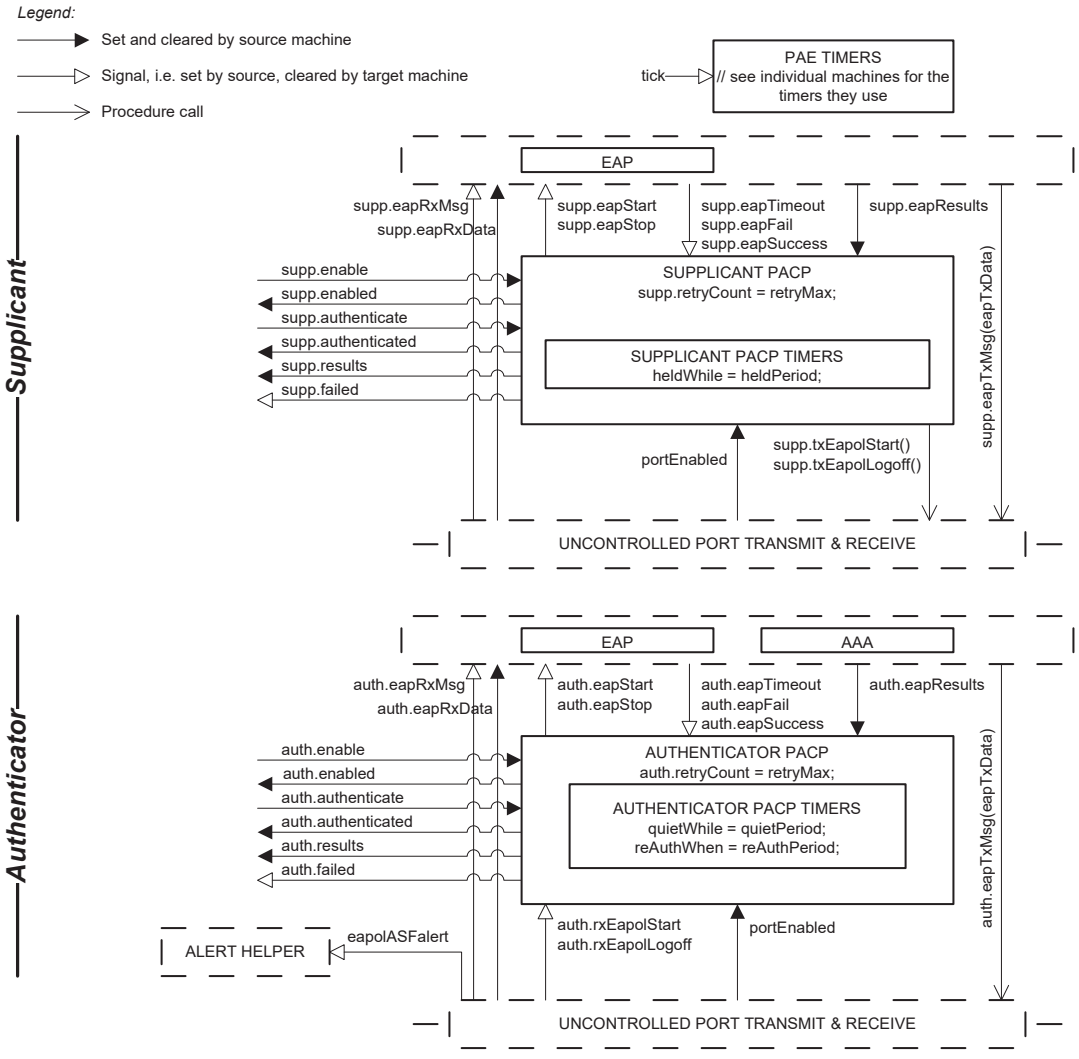


Figure 8-4—PAE state machines and interfaces

8.5 EAPOL transmit and receive

The following variables and procedures compose the interface provided by the EAPOL transmit and receive functions that support the PAE.

— portEnabled: Set by the EAPOL entity if EAPOL PDUs can be transmitted and received by the PAE. Where the Port is supported directly by a LAN MAC, or an equivalent service that support the MAC Internal Sublayer Service (ISS), portEnabled is set if MAC_Operational is TRUE. Where the Port is supported by the EAPOL functions that create virtual ports dynamically, those functions control portEnabled.

— eapRxMsg, eapRxData, eapTxMsg(eapTxData): As specified in 8.3.

— rxEapolStart: Set by the EAPOL entity on receipt of an EAPOL-Start if the PAE is an Authenticator, cleared by PACP.

— rxEapolLogoff: Set by the EAPOL entity on receipt of an EAPOL-Logoff if the PAE is an Authenticator, cleared by PACP. A received EAPOL-Logoff may be discarded without setting rxEapolLogoff if connectivity resulting from authentication is (or will be) secured by MACsec or IEEE 802.11 procedures (6.6), or supervised by MKA.

— txEapolStart(): Procedure called by a Supplicant PAE's PACP entity to transmit an EAPOL-Start.

— txEapolLogoff(): Procedure called by a Supplicant PAE's PACP entity to transmit an EAPOL-Logoff. If connectivity resulting from authentication is (or will be) secured by MACsec or IEEE 802.11 procedures (6.6), or supervised by MKA, this procedure may return without transmitting an EAPOL-Logoff.

8.6 Supplicant and Authenticator PAE timers

This standard adopts the convention of defining state machine timers as variables that are decremented, if their value is nonzero, by the operation of a state machine each time a variable tick is set. The system environment is responsible for setting this variable for each timer state machine at regular intervals. All timers used by the PAE state machines have a resolution of one second; i.e., the initial values used to start the timers are integer values, and they represent the timer period as an integral number of seconds.

NOTE—It is permissible to introduce a degree of jitter into the initialization of these timers; for example, to distribute the timing of EAPOL frame transmissions among Ports in multi-Port implementations.

The following timer is used by the Supplicant PACP state machine:

— heldWhile: Imposes a waiting period after a failed authentication attempt, before another attempt is permitted. The initial value of this timer, heldPeriod, may be read and may be set, to any value in the range from 0 to 65 535 s by management. Its default value is 60 s.

The following timers are used by the Authenticator PACP state machine:

— quietWhile: Imposes a waiting period after a failed authentication attempt, before another attempt is permitted. The initial value of this timer, quietPeriod, may be read and may be set, to any value in the range from 0 to 65 535 s, by management. Its default value is 60 s.

— reAuthWhen: Determines when reauthentication of the Supplicant takes place. The initial value of this timer, reAuthPeriod, may be read and may be set by management. Its default value is 3600 s.

These timers are decremented by the Supplicant Timers and Authenticator Timer state machines respectively (Figure 8-5).

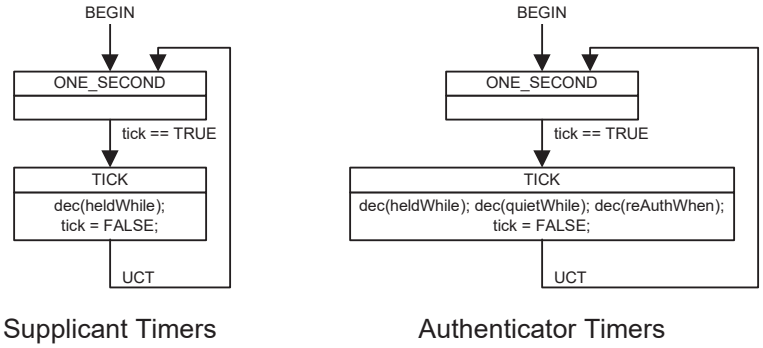


Figure 8-5—PAE Timer state machines

8.7 Supplicant PACP state machine, variables, and procedures

The Supplicant PACP shall implement the state machine in Figure 8-6; state machine variables and procedures are those defined for the higher layer (8.3), client (8.4), and EAPOL transmission and reception (8.5) interfaces together with the following:

— retryCount: Count of the number of authentication attempts.

— retryMax: Maximum number of attempts before failure is reported to the Logon Process, and the heldWhile timer imposed before further attempts are permitted. This parameter is not modified by the state machines but may be read and may be set by management, with default value is 2.

8.8 Supplicant PAE counters

The following counters may be maintained by the Supplicant PAE state machines for diagnostic purposes. Each counts the number of times that the specified state transition, action, or event occurs.

— **suppEntersAuthenticating**:
Transitions to AUTHENTICATING from UNAUTHENTICATED.

— **suppAuthTimeoutsWhileAuthenticating**:
supp.eapTimeout becomes TRUE in the AUTHENTICATING, re-entering that state (if the retry count has not be exceeded) or a transition to UNAUTHENTICATED.

— **suppEapLogoffWhileAuthenticating**:
Transitions to LOGOFF, invoking supp.txEapolLogoff(), from AUTHENTICATING.

— **suppAuthFailWhileAuthenticating**: Transitions from AUTHENTICATING to HELD.

— **suppAuthSuccessesWhileAuthenticating**: Transitions to the AUTHENTICATED state.

— **suppAuthFailWhileAuthenticated**:
Transitions to AUTHENTICATING from AUTHENTICATED.

— **suppAuthEapLogoffWhileAuthenticated**:
Transitions to LOGOFF, invoking supp.txEapolLogoff(), from AUTHENTICATED.

NOTE 1—For clarity, the actions involved in incrementing these counters are not shown on the accompanying state machines, as including all the necessary actions would necessitate the introduction of additional states.

NOTE 2—These counters mirror those available for the Authenticator (8.10), and are similarly omitted from the PAE MIB specified in this revision of this standard.

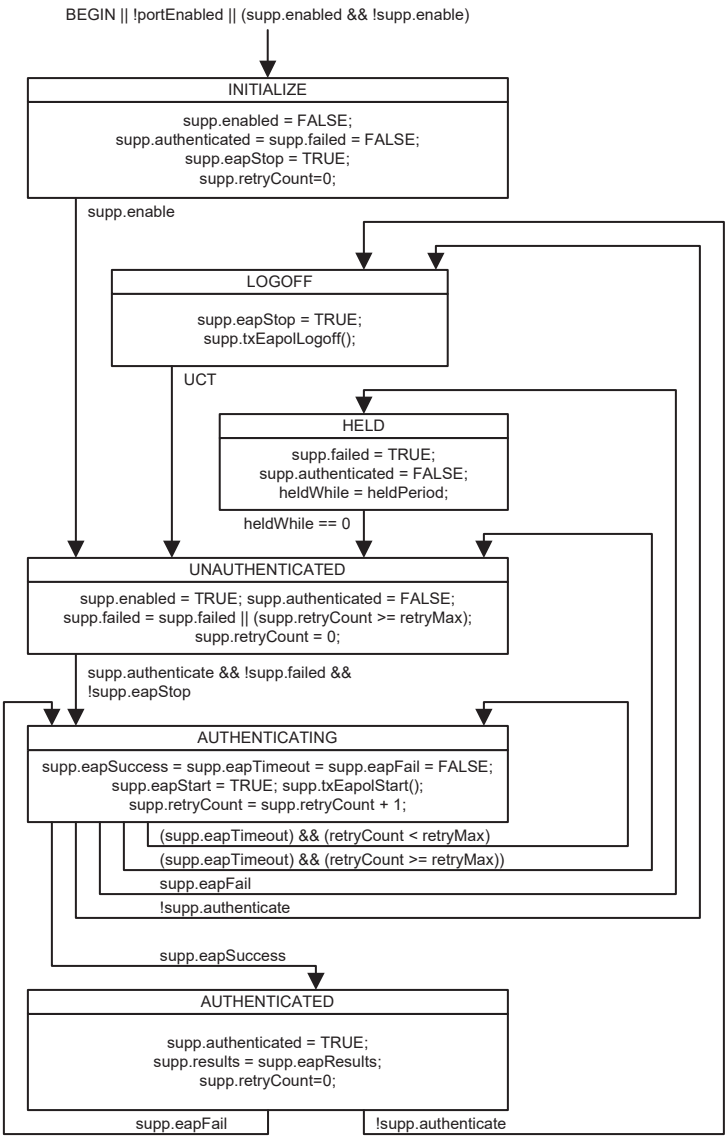


Figure 8-6—Supplicant PACP state machine

8.9 Authenticator PACP state machine, variables, and procedures

The Authenticator PACP shall implement the state machine in Figure 8-7; state machine variables and procedures are those defined for the higher layer (8.3), client (8.4), and EAPOL transmission and reception (8.5) interfaces together with the following:

— retryCount: Count of the number of authentication attempts.

— retryMax: Maximum number of authentication attempts before failure is reported to the Logon Process, and the quietWhile timer imposed before further attempts are permitted. This parameter is not modified by the state machines but can be changed by management, and has default value 2.

— reAuthEnabled: TRUE if PACP should initiate reauthentication periodically, FALSE otherwise. The default value of this parameter, which is not modified by the state machines but can be changed by management, is FALSE.

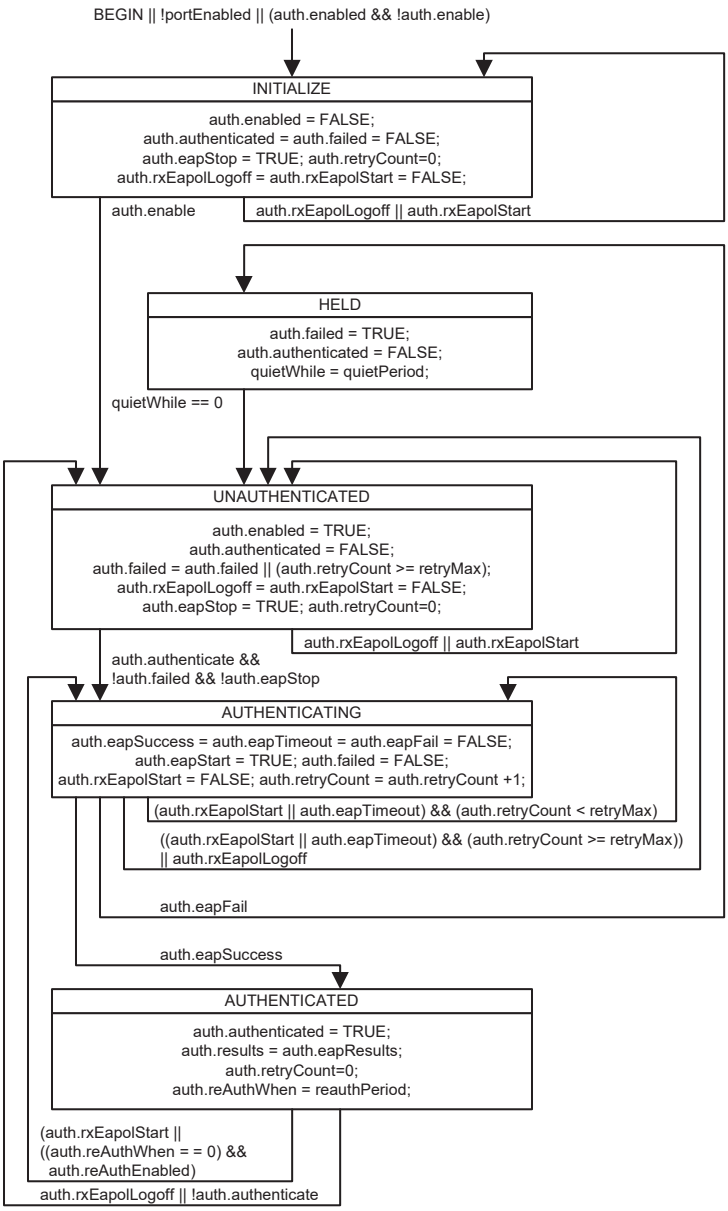


Figure 8-7—Authenticator PACP state machine

8.10 Authenticator PAE counters

The following counters may be maintained by the Authenticator PAE state machines for diagnostic purposes. Each counts the number of times that the specified state transition, action, or event occurs.

— **authEntersAuthenticating**: Transitions to AUTHENTICATING from UNAUTHENTICATED.

— **authAuthTimeoutsWhileAuthenticating**: auth.eapTimeout becomes TRUE in the AUTHENTICATING state, resulting in either re-entry to that state (if the retry count has not be exceeded) or a transition to UNAUTHENTICATED.

— **authAuthEapStartsWhileAuthenticating**: auth.rxEapolStart becomes TRUE in the AUTHENTICATING state, resulting in either re-entry to that state (if the retry count has not be exceeded) or a transition to UNAUTHENTICATED.

— **authAuthEapLogoffWhileAuthenticating**: auth.rxEapolLogoff becomes TRUE in the AUTHENTICATING state, resulting in a transition to UNAUTHENTICATED.
— **authAuthSuccessesWhileAuthenticating**: Transitions to the AUTHENTICATED state.
— **authAuthFailWhileAuthenticating**: Transitions from AUTHENTICATING to HELD.
— **authAuthReauthsWhileAuthenticated**: auth.rxEapolStart becomes TRUE in the AUTHENTICATED state, resulting in a transition to AUTHENTICATING.
— **authAuthEapStartsWhileAuthenticated**: auth.reAuthWhen timer expires in the AUTHENTICATED with auth.reAuthEnabled TRUE, resulting in a transition to AUTHENTICATING.
— **authAuthEapLogoffWhileAuthenticated**: Transitions from AUTHENTICATED to UNAUTHENTICATED.

NOTE 1—For clarity, the actions involved in incrementing these counters are not shown on the accompanying state machines, as including all the necessary actions would necessitate the introduction of additional states.

NOTE 2—These counters were initially included in the PAE MIB, but were deprecated in IEEE Std 802.1X-2004 and are omitted from the PAE MIB specified in this revision of this standard.

8.11 EAP methods

EAP (IEFT RFC 3748 [B14]) provides a generic authentication framework that supports many different types of authentication methods with different characteristics. Section 7.2.1 of IETF RFC 3748 [B14] defines some important security claims that can be made for particular methods. All EAP methods used by a PAE shall support mutual authentication.

A passive adversary between a Supplicant and EAP authenticator can observe any information that an EAP method passes without confidentiality protection. This could be considered a privacy threat to the Supplicant. Some EAP methods [e.g., a “tunneled EAP” method such as TEAP (IETF RFC 7170)] protect the complete contents of the authentication process from a passive adversary.

8.11.1 MKA and EAP methods

Each EAP method that is used with MKA shall

— Support key derivation. The strength of the derived keys should be at least equivalent to 128 bits to take full advantage of the security offered by MACsec.
— Generate an MSK of at least 64 octets, as required by Section 7.10 of IETF RFC 3748 [B14], of which the first 16 or 32 octets are used by this standard as described in 6.2.2.
— Generate a Session-Id as defined in IETF RFC 5247.

When used with MKA the EAP method used should have the following characteristics:

— Integrity protection
— Replay protection
— Dictionary attack protection
— Cryptographic binding
— Session independence
— Fragmentation
— Ciphersuite negotiation

and may also provide the following:

— Confidentiality
— Fast reconnect
— Channel binding

NOTE 2—EAP method requirements for Wireless LANs are specified in IETF RFC 4017 [B15].

8.11.2 Integration with IEEE Std 802.1AR and EAP methods

An implementation that claims to integrate the use of IEEE Std 802.1AR Secure Device Identifier with this standard shall implement EAP-TLS (IETF RFC 5216). That EAP-TLS implementation shall support the ciphersuite recommendations made by IEEE Std 802.1AR for integration with this standard and shall be capable of TLS version 1.2 as specified by IETF RFC 8446.

9. MACsec Key Agreement protocol (MKA)

The MACsec Key Agreement (MKA) protocol allows PAEs (6.3), each associated with a Port that is an authenticated member of a secure connectivity association (CA) or a potential CA, to discover other PAEs attached to the same LAN, to confirm mutual possession of a CAK and hence to prove a past mutual authentication, to agree the secret keys (SAKs) used by MACsec for symmetric shared key cryptography, and to ensure that the data protected by MACsec has not been delayed.

MKA comprises a secure fully distributed multipoint-to-multipoint transport and a number of applications of that transport, including the distribution of SAKs by an elected key server using AES Key Wrap. The transport allows each PAE, possessing the secret key (CAK, 6.2) that is the token of its authentication and authorization to participate in a CA, to ensure that each received protocol message has been transmitted by another participant in current possession of that key and has not been delayed or replayed. The key distribution application ensures that no participant unwittingly reuses an SAK-nonce pair, even though that participant might have been powered-off, re-initialized, or making use of another key server as the consequence of temporary LAN partition. While both the transport and key distribution support group CAs, providing continuous secured connectivity even as the participants in the CA change, they impose little overhead where the number of group participants is two as compared to protocols designed solely for pairwise key agreement, and degenerate to well known forms in that common case.

In addition to distributing fresh SAKs, MKA manages their installation and use by the SecYs that secure the data transmitted and received by each Controlled Port, ensuring that each is capable of receiving data protected by that SAK before it is used for transmission. Each SecY uses MKA to communicate the lowest PN used for transmission with the SAK within the last two seconds, allowing receivers to bound transmission delays.

The Key Agreement Entity (KaY) responsible for MKA within the PAE can operate multiple simultaneous MKA instances: allowing CAK changes while maintaining secured connectivity; allowing confirmation of the hints provided by network discovery before a particular CA and CAK is selected; and allowing the key server to use a number of initial CAKs, each providing pairwise authentication between the server and a member of a group CA, to distribute the CAK for that group.

This clause specifies the following:

a) Design requirements for MKA protocol functionality, including security considerations (9.1).
b) Operational requirements that MKA places on the system and on network administration (9.2).
c) The key hierarchy used by MKA (9.3).
d) The secure transport provided by MKA to its applications (9.4).
e) Key server election, using the secure transport to exchange information (9.5).
f) How MKA capable PAEs decide to use, or not use, MACsec (9.6).
g) Cipher suite selection (9.7).
h) SAK generation, distribution, and adoption (9.8).
i) Assignment of SAKs to SAIs (9.9).
j) SAK installation and use (9.10).
k) Detection of changes in the secured connectivity provided by each Controlled Port (9.11).
l) Group CAK distribution (9.12).
m) Secure transmission of announcement and other parameters to provide confirmation of values previously received in unprotected EAPOL frames (9.13).
n) Creation and deletion of MKA participants (9.14).
o) Management of the KaY and MKA (9.16).
p) Temporary suspension of MKA operation to facilitate in-service control plane software upgrades without disrupting existing secure connectivity (9.18).

The following terms are used to identify roles within the protocol or protocol scenarios:

— **participant:** The personification of a single KaY's participation in a given MKA instance (i.e., transmitting and receiving MKPDUs protected by keys derived from a single given CAK and identified by a given CKN), operating with positive intent, and obeying the protocol.

— **actor:** The participant under discussion, usually in the KaY being described.

— **partners:** Other participants in the same MKA instance, and attached to the same LAN, as the actor.

— **successful actor:** An actor that has one or more live partners and is participating in an MKA instance that has elected a Key Server.

— **principal actor:** The successful actor selected by a KaY to control its associated PAC or SecY.

— **member:** The personification of a single KaY's participation in all MKA instances and use of other controls that determine the connectivity provided by its associated Controlled Port. A system is described as being member of a CA if it includes a Controlled Port providing connectivity using that CA.

Throughout this standard 'random' and 'randomly chosen' numbers meet the criteria specified in 9.2.

9.1 Protocol design requirements

MKA meets the following requirements:

a) It meets its security and other goals, detailed in these requirements, in the presence of an attacker that does not possess the CAK but can copy any frame sent by any participant, can selectively prevent delivery to some participants, and can transmit arbitrary frames to arbitrary participants.

b) It provides and refreshes keys and other information that its clients (KaYs) use to establish, maintain, and manage secure connectivity in the presence of an attacker that does not possess the CAK but can copy frames transmitted by any participant, and can transmit arbitrary frames to arbitrary participants, up to the point that the load imposed by the reception of those frames exceeds the resources of a participant.

c) Following any period of 8 seconds during which all frames transmitted by each of a set of participants are delivered, once and without misordering, to each of the other participants, and the load imposed by frames received from an attacker does not exceed the resources of any participant, MKA will provide the keys and information required by each of its participants' clients, irrespective of the prior state of each participant or frames buffered by the LAN.

   NOTE 1—The requirement is that of correctness: the protocol convergence time is bounded; rather than a performance goal. The figure of 8 seconds arises from the possibility that prior participants are being timed out (over a period of MKA Life Time, 6 seconds, see Table 9-1) just as a new participant joins, plus MKA Hello Time (2 seconds) to ensure that all participants have subsequently transmitted.

   NOTE 2—If MKA operation is suspended (9.18), and the participants do not already possess the necessary keys and information, convergence will be delayed until MKA operation resumes.

d) It requires minimal transmission bandwidth, with a constant number of frames per second per participant in steady operation up to the MKA design goal of 30 participants.

e) It does not require its participants to retain any information from prior instantiations of the protocol, with the exception of the CAK and CKN.

f) It provides mutual discovery of the potential members of a CA, i.e., systems that possess the same CAK identified by the same CKN, that are attached to the same LAN.

g) It provides proof of mutual liveness, i.e., mutual proof of current possession of the CAK, between all pairs of CA members.

h) When used in combination with a SecY, with strict validation of received MACsec frames as specified by IEEE Std 802.1AE, it provides a reliable indication of the operPointToPointMAC status parameter.

i) It detects duplication of the individual MAC addresses used to compose SCIs.

j) It can be used to support mutual identification of participants that have previously completed mutual authentication, but are not capable of operating MACsec.

k) It allows its participants to ensure that the data frames protected by MACsec are not being delayed by more than 2 seconds.

NOTE 3—Delay protection guards against an attack on the configuration protocols that MACsec is designed to protect by alternately delaying and delivering their PDUs. Delay protection does not operate if and when MKA operation is suspended (9.18).

MKA does not, is not capable of, or makes no attempt to defend against byzantine attacks, i.e., attacks made by systems that possess the CAK, including the following that are specifically identified as non-goals:

l) Deliberate redistribution of a key by a participant acting as a key server.

MKA satisfies its security and convergence requirements to within the probabilities set by the use of pseudo random numbers generated as specified in 9.2.

When the option to support MACsec Cipher Suites that use Extended Packet Numbering is implemented, MKA meets the following requirements:

m) Recovery, if lost, of the 32 most significant bits of the XPN, in support of the protocol correctness requirement c) above.

9.2 Protocol support requirements

A system that implements MKA shall satisfy this standard’s requirements for the following:

a) Random number generation

b) SC identification

A system that implements MKA in support of MACsec shall also meet the protocol support requirements specified in IEEE Std 802.1AE.

9.2.1 Random number generation

The system shall provide a strong random number generator (RNG) for the generation of all numbers described as random in this standard. If a non-deterministic RNG (e.g., hardware RNG) is not available, the system shall make use of sufficient entropy to create a good quality seed for a deterministic RNG and should conform to the requirements of NIST Special Publication 800-108.

9.2.2 SC identification

Each SC is identified by an SCI that comprises a MAC address and a Port Identifier, unique within the system that has been allocated that address.

9.3 MKA key hierarchy

The root of key hierarchy for any given instance of MKA is the secure Connectivity Association Key (CAK), a secret key. Possession of a CAK for the CA is a prerequisite for membership in each CA supported by MACsec, and all potential members possess the same CAK and are attached to the same LAN.

The overall key hierarchy used by this standard is specified in 6.2, including the distribution of the CAK to, or its acquisition by, each potential member of a given CA and its relationship to authentication protocols. In summary, a CAK can be a direct result of a system's participation in EAP (Clause 8), a pre-shared key (PSK), or a key chosen by an MKA key server and distributed using a prior MKA instance. This subclause (9.3) specifies the following:

a) How CAKs and derived keys are identified (9.3.1) within MKA.

b) Independence of each CAK from its predecessors and successors (9.3.2).

c) The derivation of further secret keys from the CAK (9.3.3) including the following:

   1) ICK, the ICV key used to verify the integrity of MKPDUs

   2) KEK, the key encrypting key used together with AES Key Wrap to distribute SAKs

9.3.1 CAK identification

Each CAK is identified by secure Connectivity Association Key Name (CKN) that allows each of the MKA participants to select which CAK, or CAK derived key, to use to process a received MKPDU. MKA places no restriction on the format of the CKN, save that it comprise an integral number of octets, between 1 and 32 (inclusive), and that all potential members of the CA use the same CKN. No further constraints are placed on the CKNs used with PSKs, though the network administrator should be aware that accidental use of the same CKN for different CAKs can inhibit communication and cause an operator to falsely conclude that a security attack is being attempted. CKNs for group CAKs that are generated by a Key Server shall include an RNG of at least 128 bits to guard against CKN collisions. See 6.2 for EAP specific CKN recommendations.

9.3.2 CAK Independence

A CA can be extremely long-lived, in infrastructure applications (see Clause 7) a design goal of tens of years without significant interruption in communication is not unrealistic, and it can be expected that systems supporting the CA (e.g., authentication servers) and even equipment directly participating in the CA will be changed during that time. MKA allows the communication between the members of a CA to be supported by a succession of MKA instances, as specified in 12.4, without interruption. While continuing communication while the CAK is changed necessarily implies that all of the systems that are members of the CA both before and after the change know both CAs, the two MKA instances each maintain their own CAK and derived keys quite separately, and neither instance discloses its CAK or derived keys for the other to its participants.

NOTE—Opinions differ as to how long an interruption in communication can last before it is considered significant, typically ranging from a few milliseconds to a few seconds. The goal of MKA, and its use in this standard, is to allow CAKs to be changed without any interruption to MACsec protected communication.

9.3.3 Derived keys

Each of the keys used by MKA is derived from the CAK. The CAK is not used directly. The derived keys are tied to the identity of the CAK, and thus restricted to use with that particular CAK.

To accommodate future developments in cryptography, each MKPDU conveys an Algorithm Agility parameter that identifies how the ICK is derived from the CAK, and how it is used. Each Algorithm Agility parameter value comprises four octets, the first three being those of an OUI (Organizationally Unique Identifier) or CID (Company ID) allocated by the IEEE Registration Authority, and the fourth allocated by the organization to which that OUI or CID has been allocated. The derivation of the ICK, and the use of this key as specified in this standard, is identified by the value specified in Table 9-1.

Table 9-1—MKA Algorithm Agility parameter values

Parameter value	Specification
00-80-C2-01	IEEE Std 802.1X-2020

The ICK is derived from the CAK using the KDF specified in 6.2.1. This KDF uses the AES Cipher in CMAC mode (IETF RFC 4493). The ICK is derived from the CAK using the following transform:

ICK = KDF(Key, Label, Keyid, ICKLength)

where

Key = CAK

Label = "IEEE8021 ICK"

Keyid = the first 16 octets of the CKN, with null octets appended to pad to 16 octets if necessary

ICKLength = two octets representing an integer value (128 for a 128 bit ICK, 256 for a 256 bit ICK) with the most significant octet first

The Label is a UTF-8 string, without a null or other termination, exactly 12 bytes in length (the quotes shown do not form part of the string and exactly one space separates '8021' and 'ICK'). The length of the Label is chosen to make the input to the PRF within the KDF exactly 32 bytes. The hexadecimal representations of each of the text strings used with this KDF and test vectors are given in Annex G.

To accommodate future developments in cryptography, each of the MKA parameter set types used by MKA to communicate a wrapped key identifies the Key Wrap Algorithm and KEK derivation (6.2.4, Figure 11-11, Figure 11-12, Figure 11-13) used. The KEK used by all parameter set types currently specified in this standard is derived from the CAK using the KDF specified in 6.2.1. This KDF uses the AES Cipher in CMAC mode (IETF RFC 4493). The KEK is derived from the CAK using the following transform:

KEK = KDF(Key, Label, Keyid, KEKLength)

where

Key = CAK

Label = "IEEE8021 KEK"

Keyid = the first 16 octets of the CKN, with null octets appended to pad to 16 octets if necessary

KEKLength = two octets representing an integer value (128 for a 128 bit KEK, 256 for a 256 bit KEK) with the most significant octet first

The Label is a UTF-8 string, without a null or other termination, exactly 12 bytes in length (the quotes shown do not form part of the string and exactly one space separates '8021' and 'KEK'). The length of the Label is chosen to make the input to the PRF within the KDF exactly 32 bytes.

When SAKs and CAKs are distributed they are protected by the KEK, using an AES Key Wrap as defined in 9.8.2 and 9.12.1.

SAKs should be derived from the CAK as specified in 9.8.1, but may also be generated directly by the Key Server's strong random number generator (RNG, 9.2.1). Distributed CAKs, when used, shall be random values generated by the MKA Key Server RNG. Each distributed CAK is distinct from previously distributed CAKs, so that an MKA participant or attacker holding only the current CAK cannot determine a previously distributed CAK. This allows implementation of a policy of perfect forward security, with a fresh CAK being distributed when each participant joins a CA, so that participant cannot decrypt wrapped keys from previously transmitted MKA frames.

NOTE—MKA does not require fresh CAK distribution when a new participant joins a CA, as that would prolong the process of joining.

9.4 MKA transport

MKA provides a secure multipoint-to-multipoint transport between the members of the same CA, suitable for conveying information that is constant, or refreshed or acknowledged by the MKA applications that make use of that transport. The CAK is used to authenticate each protocol data unit (MKPDU) transmitted, providing proof of its transmission by a CA member, and each station includes its own randomly chosen identifier and a message number in the MKPDU. By transmitting MKPDUs that contain the identifiers and recent message numbers of the other participants, each member proves that it is in current possession of the CAK and is actively participating in the protocol, thus demonstrating the 'liveness' of the MKPDU and distinguishing it from MKPDUs that could have been captured by an attacker and played or replayed later—with the aim of disrupting the protocol or of influencing its outcome. MKPDUs are transmitted at regular intervals (see Table 9-3) of MKA Hello Time or MKA Bounded Hello Time (if a bounded receive delay is to be guaranteed, see 9.10), when data to be transported changes as specified in this clause (Clause 9), and as specified by the CP state machine (setting the state machine variable newInfo, as specified in Clause 12).

The message numbers also serve to enforce in-order delivery, and each of the MKA applications is designed so that the information conveyed in each MKPDU is idempotent, i.e., can be repeated without further changing the state of a recipient, and complete, i.e., fully expresses the desire of the transmitter for state change at the recipient. This design philosophy simplifies protocol analysis and allows a receiver to discard MKPDUs with prior message numbers.

The MKA transport is fully distributed and, as a consequence, robust in the face of the failure of any participant or of the LAN connectivity to that participant.

In principle the MKA transport could be used to communicate its participants' desires to cooperate in a range of applications that require mutual authentication, and to state the relevant capabilities and requirements for each, since the MKPDU format and interoperable version rules permit the addition of parameter sets. In practice MKA is focused on MACsec. The generality provided by the MKA transport and MKPDU does permit future standardization of support for additional applications, alternatives for or additions to MACsec, and extensions and refinements for cipher suite negotiation. An example of MKA's extensibility is provided by its use to secure announcements (9.13).

9.4.1 Message authentication

Each protocol data unit (MKPDU) transmitted is integrity protected by an 128 bit ICV, generated by AES-CMAC using the ICK (9.3):

ICV = AES-CMAC(ICK, M, 128)

M = DA + SA + (MSDU – ICV)

In other words, M comprises the concatenation of the destination and source MAC addresses, each represented by a sequence of 6 octets in canonical format order, with the MSDU (MAC Service Data Unit) of the MKPDU including the allocated Ethertype, and up to but not including, the generated ICV.

NOTE—M comprises the whole of what is often referred to as 'the frame' considered from the point of view of the MAC Service provided by Common Port of the SecY (Figure 6-2) or PAC (Figure 6-6) supporting MKPDU transmission. The description does not use the term 'frame', because that Common Port could be supported by additional VLAN tags or other tags (consider the upper SecY shown in Figure 7-17) prior to transmission of a MAC frame by a system. Any such additional tags would not be covered by the ICV, and would be removed prior to MKPDU reception by a peer PAE.

Since the ICK is not directly distributed by any protocol, but only derived from the CAK, verification of the ICV both ensures that the contents of the MKPDU have not been modified but also that it was composed by a system that possessed the CAK.

Each MKPDU contains a secure Connectivity Association Key Name (CKN, 9.3.1) so intended recipients can identify the CAK, and hence the ICK, to be used to verify the ICV. The Algorithm Agility parameter conveyed in each MKPDU (9.3.3) permits future specification of other ICV algorithms and sizes.

9.4.2 Member identification and message numbers

Each participant in the protocol chooses a random (9.2) 96-bit member identifier (MI) when its participation in the protocol begins, and this MI is used, together with a 32-bit message number (MN) that is initialized to 1 and incremented with each MKPDU transmitted, to protect against delayed transmission or replay. Once a receiver has authenticated (9.4.1) an MKPDU with a given MI and MN, any MKPDUs with the same MI and the same or lower MN are discarded.

Use of the randomly chosen MI rather than a permanent unique identifier such as a MAC Address as the base for the anti-delay and anti-replay functions provided by the MN removes the need for a participant to maintain a record of the last used message number when powered off or re-initialized or for that participant to attempt to recover that data from its peers when an attacker has to be assumed to be active. Use of the MI also solves the issue of new participants understanding a potentially wrapping sequence number—when the MN reaches its upper limit the participant simply chooses a new MI and restarts the MN at 1.

While the MKPDUs themselves do not rely on the uniqueness of a key-nonce pair to ensure confidentiality or to avoid to disclosing data that might lead to the recovery of a key, a MAC address forms part of the SCI which is a component of the nonce used by MACsec. Use of the MI makes it clear to any observer when, as rarely but occasionally happens, a duplicate MAC address is in use, either accidentally or as part of an attack on MACsec. The KaY will not enable MACsec transmission if there is any risk of a duplicate SCI.

The participant chooses a new MI if its current MI is already in use, as detected by either of the following:

a) Its presence in the MI field of a received MKPDU.

b) Its use in the Live Peer List or Potential Peer List of a received MKPDU in combination with an MN that is greater than that last transmitted by the participant.

9.4.3 Determining liveness

A participant proves liveness to each of its peers by including their MI, together with an acceptably recent MN, in an MKPDU with the participant's own MI and MN.

To avoid a new participant having to respond to each MKPDU from each partner as it is received, or trying to delay its reply until it is likely that MI.MN tuples have been received from all potential partners, each participant maintains and advertises both a Live Peer List and a Potential Peer List. The Live Peer List includes all the peers that have included the participant's MI and a recent MN in a recent MKPDU, and the Potential Peer List includes all the other peers that have transmitted an MKPDU that has been directly received by the participant or that were included in the Live Peer List of a MKPDU transmitted by a peer that has proved liveness. Peers are removed from each list when an interval of between MKA Life Time (see Table 9-3) and MKA Life Time plus MKA Hello Time has elapsed since the participant's recent MN (see above) was transmitted. This time is sufficient to ensure that two or more MKPDUs will have been lost or delayed prior to the incorrect removal of a live peer.

NOTE 1—The specified use of the Live and Potential Peer Lists thus permits rapid removal of participants that are no longer active or attached to the LAN while reducing the number of MKPDUs transmitted during group formation. For example, a new participant will be admitted to an established group after receiving, then transmitting, one MKPDU.

NOTE 2—A suspended participant (9.18) will be removed from the Live and Potential Peer Lists as described, but its associated SecY will still be able to transmit and receive secure frames until other CA members adopt a new SAK.

9.4.4 MKPDU information elements and application data

The MKA transport encodes the following information in each MKPDU:

a) The CA Key Name

b) The transmitting participant's SCI

c) The transmitting participant's MI and MN

d) The Live Peer List, comprising the MI and MN of each of the other participants believed to be in current possession of the CAK

e) The Potential Peer List, comprising the MI and MN of each of the other potential participants

For encoding and decoding convenience, these elements are not consecutively encoded, in particular the live and potential peer lists are encoded after the information provided by the MKA applications. The details of MKPDU encoding and preliminary validation of received MKPDUs are specified in Clause 11.

9.4.5 Addressing

A LAN can be an individual LAN, bounded by the extent of its supporting media access method and media access method procedures, or can be supported by bridges in a Bridged Local Area Network or Virtual Bridged Local Area Network. Interoperability, avoiding redundancy, protecting the infrastructure, and the need to support communication between stations in the network, all necessitate placing restrictions on where, within the interface stack that composes a port (IEEE Std 802.1AC), SecYs are placed. Clause 11 of IEEE Std 802.1AE-2006 specifies the use of MACsec within systems, and those restrictions, but explicitly recognizes that MACsec can be used across, within, and to secure access to a Provider Bridged Network. The destination address used by MKA has to be aligned with the placement of port-based network access control in the interface stack and the corresponding use of MACsec within the protocol stack, and shall be a group address. Table 11-1 specifies group addresses that support the application scenarios described in this standard (Clause 7).

NOTE—Use of a group address that is filtered by bridges and the inclusion of destination and source MAC addresses within the ICV calculation makes attacking MKA from a distance more difficult.

The source address of each MKPDU shall be an individual MAC Address assigned to the port transmitting that MKPDU.

9.4.6 Active and passive participants

A participant can be active, transmitting periodic MKPDUs, or passive. A passive participant will become active for a period of MKA Lifetime following the receipt of an MKPDU from a feasible partner, i.e., provided that either the receiving participant or the partner is prepared to act as a Key Server. Whether a participant is to be active when first created, and whether it is to remain active in the absence of feasible partners depends on the port-based network access control application. The creation of passive participants supports systems that have many potential peers, with only one or a few likely to be connected at a time. Participants that are always active are desirable where connectivity is provided by media that do not reliably signal loss and resumption of connectivity, as can be the case for infrastructure links supported by virtual media. If all the participants in a potential CA can be passive, and an extended and undetected network outage occurs, it is possible that the potential CA members will fail to transmit MKPDUs, resulting in a permanent lack of connectivity.

NOTE 1—The condition of having recently received an MKPDU from a feasible partner can be determined by inspecting the participant's Live Peer List and Potential Peer List.

NOTE 2—The model of Logon Process operation encompasses participant creation, deletion, and control of active or passive participation (12.2, 12.5.2). The CAK cache provides management (see activate in 9.16).

An active participant will remain active, even in the absence of received MKPDUs, while a suspension is in progress (provided that the participant is not itself suspended). A suspended participant will resume active operation if and only if

a) Its CAK was previously cached, and
b) The management controls associated with that cached CAK specify that it is to be active on resumption (9.16).

9.5 Key server election

The participants in a given MKA instance agree on a Key Server, responsible for the following:

a) Deciding on the use of MACsec (9.6)
b) Cipher suite selection (9.7)
c) SAK generation and distribution (9.8)
d) SA assignment (9.9)
e) Identifying the CA when two or more CAs merge (9.11)
f) CA formation and group CAK distribution (9.12)
g) Initiating, continuing, and terminating MKA suspension (9.18)

If the CAK is a pairwise CAK derived directly from EAP (see 6.2.2), the MKA participant for the PAE that was the EAP Authenticator will be the Key Server, and will not accept information [a) through g) above] from any other participant that attempts to act as the Key Server for that MKA instance. Each participant in an MKA instance with a CAK that was not directly derived from EAP uses the Key Server Priority (an 8-bit integer) encoded in each MKPDU to agree on the Key Server. Each such participant selects the live participant advertising the highest priority as its Key Server whenever the Live Peer List changes, provided that highest priority participant has not selected another as its Key Server or is unwilling to act as the Key Server [as indicated by 9.5.1 b)]. If a Key Server cannot be selected SAKs are not distributed. In the event of a tie for highest priority Key Server, the member with the highest priority SCI is chosen. For consistency with other uses of the SCI's MAC Address component as a priority, numerically lower values of the Key Server Priority and SCI are accorded the highest priority. Table 9-2 contains recommendations for the use of priority values for various system roles. Participants that will never act as a Key Server should advertise priority 0xFF.

Table 9-2—Key Server Priority values

System role of network access controlled port	Recommended or default value	Recommended range
Infrastructure port	0x10	0x00 – 0x1F
Primary Network Access Point	0x30	0x20 – 0x3F
Secondary or backup Network Access Point	0x50	0x40 – 0x5F
Group CA member	0x70	0x60 – 0x7F

As CA membership changes and before all current participants recognize each other as live, more than one participant can decide that it is the Key Server and attempt to fulfill the Key Server's responsibilities. The design of the SAK distribution and other applications where the Key Server plays a role is such that each member can simply accept data from the Key Server that member has elected.

If a KaY participates in multiple MKA instances so that there are several actors—one per instance—for a given port, then only the actor selected as the principal actor (12.1) will (if elected Key Server for its CAK) distribute SAKs. Since the factors that cause a principal actor to be selected from its peers are the same for different CAK, CKN tuples with the same distribution, replacement of a principal actor by its successor will

occur, whenever possible, without a change of Key Server. This succession plan ensures that the SAKs distributed using one CAK, CKN tuple can be followed immediately by SAKs distributed by a successor CAK, CKN without any loss of MACsec connectivity. To minimize the chance of a CA member that possesses both CAKs temporarily losing connectivity, a Key Server should not distribute an SAK using the new CAK until MKA Life Time (Table 9-3) has elapsed after it has started participating with that CAK, and should not delete the participant for the prior CAK until MKA Life Time has elapsed after that new SAK is first distributed. The CP state machine (Figure 12-2) ensures that a new SAK is not distributed until the Key Server is receiving and transmitting using a single SAK.

NOTE 1—If two PAEs are each capable of acting as both an EAP Authenticator and an EAP Supplicant, their interaction can result in two instances of successful mutual authentication with each acting as the Authenticator in one. Thus two MKA instances can be created, and each PAE's KaY will be elected Key Server for one instance. However, each KaY will select, as its principal actor, its participant in the MKA instance with the highest priority Key Server. That Key Server can then use that MKA instance to distribute SAKs (or select unauthenticated connectivity, see 12.3), and the participants in the other MKA instance can be deleted.

NOTE 2—A number of KaYs participating in multiple MKA instances will still succeed in configuring a single CA even if each participates in a different subset of those instances, provided that the highest priority Key Server in each subset itself participates in an MKA instance with a higher priority Key Server or is the highest priority Key Server. For example, assume KaYs A, B, C, D (say) in decreasing priority order, with two MKA instances with participants {A, B, C} and {B, D} respectively. Although B will be elected Key Server for the {B, D} instance, it will not distribute SAKs to D, as B's principal actor will be in {A, B, C}. Thus a single CA will be created, including the ports associated with A, B, and C, but excluding D's. However, scenarios of this type are most likely to result from errors in manual key distribution. They can give rise to temporary interruptions or unwanted connectivity, particularly where in-service upgrades are to be performed and need to be eliminated as part of managing upgrades (9.18.6).

9.5.1 MKPDU application data

Each MKA participant encodes the following information in every MKPDU transmitted:

a) Key Server Priority, the priority of the transmitting participant.

b) Key Server, a flag set if and only if the participant has not decided that another participant is or will be the Key Server.

9.6 Use of MACsec

Each participant advertises whether it is capable of using MACsec, and whether it requests MACsec protection of any communication through the Controlled Port of a SecY. The Key Server decides whether all, or none, of the participants in the CA should use MACsec to protect transmit frames; thus guarding against partial connectivity within the CA.

NOTE 1—The 'MACsec desired' parameter is provided to allow a network administrator to deploy MACsec capable systems, to verify their operation—including their participation in authentication and key agreement protocols, and to ensure that network operation does not depend on communication with systems that are not MACsec capable, before MKA selects MACsec protection and restricts the CA to MACsec capable systems.

To avoid duplication or conflict with the management controls for the SecY's secure frame generation and verification (10.5 and 10.6 of IEEE Std 802.1AE-2018) direct management access to the protectFrames parameter is disabled. If the Key Server and at least one other live participant are MACsec capable, and at least one of those participants requests MACsec protection, the Key Server advertises 'MACsec protect' and each participant sets its SecY's protectFrames parameter. Otherwise, protectFrames is cleared.

NOTE 2—IEEE Std 802.1AE mandates the capability to disable management access to individual parameters.

Participants that receive or transmit unprotected frames are responsible for ensuring that their protocol clients select appropriate policies. Whether any communication is desirable depends on the system configuration (see 7.3.3). If the Key Server selects MACsec protection, any MKA participant that is not capable of using MACsec (or the selected cipher suite) will not receive or transmit using the Controlled Port, and will maintain MAC_Operational False for that port.

9.6.1 MKPDU application data

Each MKA participant encodes the following information in every MKPDU transmitted:

a) MACsec capability, indicating whether MACsec is implemented, and if so whether the implementation provides integrity protection only, integrity and integrity with confidentiality, or integrity and integrity with confidentiality with a selectable confidentiality offset of 0, 30, or 50 octets (see Table 11-6, IEEE Std 802.1AE).

NOTE—IEEE Std 802.1AE-2006 introduced the confidentiality offset to facilitate early MACsec deployment on existing systems that needed to store received frames before applying MACsec processing and that needed to examine the initial octets of received frames to decide where to store those frames. The XPN Cipher Suites standardized in IEEE Std 802.1AE do not support confidentiality offsets of other than 0.

b) MACsec desired, a flag, set if the participant desires the use of MACsec to protect frames.

9.7 Cipher suite selection

The cipher suite, if MACsec is used to protect frames, is selected by the Key Server and advertised with each key.

Cipher suites are identified by a reference number (Table 14-1 of IEEE Std 802.1AE-2018) comprising eight octets, assigned by an organization that has been authorized to use a 24-bit OUI (IEEE Std 802) or a 36-bit OUI-36 by the IEEE Registration Authority. If the assignment is OUI based, the first three octets of the reference number are the octets of the OUI with the least significant bit of the first octet zero. If the assignment is OUI-36 based, the first four octets and the four most significant bits of the fifth octet comprise the OUI-36. The remainder of the reference number is assigned by the organization. The reference number should not correspond to any other EUI-64 assignment made by the assignee, nor should the first six octets be those of any EUI-48 assignment.[18]

An MKA Key Server can also elect not to use MACsec to secure communication, using MKA simply to prove prior authentication (see 7.2). The Key Server uses an EAPOL-MKPDU to signal such use by including the Distributed SAK parameter set with a zero parameter set body length (see Figure 11-13). It is often convenient to refer to the use of plain text transmission in this context as use of the Null Cipher Suite. This standard allocates the cipher suite reference number FF-FF-FF-FF-FF-FF-FF-FF to identify plain text transmission.

9.7.1 MKPDU application data

A participant that believes itself to be the Key Server and its KaY's principal actor encodes the following information with each MACsec SAK that it distributes, unless the mandatory Default Cipher Suite GCM-AES-128 is to be used:

a) MACsec Cipher Suite, the Cipher Suite reference number.

The following information is also distributed with each MACsec SAK:

b) Confidentiality Offset, indicating whether confidentiality is to be provided, and whether an offset of 0, 30, or 50 octets is used (see IEEE Std 802.1AE).

NOTE—The XPN Cipher Suites standardized in IEEE Std 802.1AE do not support confidentiality offsets of other than 0.

If a receiving MKA participant does not implement the referenced Cipher Suite with the selected confidentiality offset, the distributed SAK will not be installed (12.4). An MKA participant should advertise

[18]See the "Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)."

any Cipher Suites implemented in addition to the Default Cipher Suite by including an Announcement parameter set (11.11.1, Figure 11-15) with a MACsec Cipher Suites TLV (11.12.3) in each MKPDU transmitted.

9.8 SAK generation, distribution, and selection

The Key Server is responsible for generating (9.8.1) and distributing MACsec SAKs, using AES Key Wrap (9.8.2), to each of the other members of the CA, using the MKA transport.

Each SAK is identified by a 128-bit Key Identifier (KI), comprising the Key Server's MI (providing the more significant bits) and a 32-bit Key Number (KN) assigned by that Key Server (sequentially, beginning with 1). Each KI is used to identify the corresponding SAK for the purposes of SAI assignment (9.9), and appears in the clear in MKPDUs, so network management equipment and personnel can observe and diagnose MKA operation (if necessary) without having access to any secret key.

NOTE 1—A KN of 0 indicates that MACsec is not being used by the transmitting participant. Use of this reserved KN value allows the participant to report that it has elected the Key Server whose MI appears in the KI.

If the Current Cipher Suite is not using extended packet numbering, the Key Server observes the Key Identifier and Lowest Acceptable PN for the most recent SAK in use, as transmitted by each CA member (the LKI and LLPN if LRX is true, and the OKI and OLPN otherwise; 9.10.1), and shall distribute a fresh SAK (subject to the constraints specified in 9.5 and this clause) if that Key Identifier matches the KI of the key most recently distributed and that Lowest Acceptable PN equals or exceeds the constant PendingPNExhaustion. PendingPNExhaustion is 0xC000 0000 for 32-bit PNs and 0xC000 0000 0000 0000 for 64-bit PNs. Subject to conditions [a) through c), below] that limit the frequency of SAK changes, postponing their generation and distribution until CA membership is likely to be stable, the Key Server shall also distribute a fresh SAK whenever a member is added to the live membership of CA (as perceived by the Key Server—with each MI, not the associated SCI, representing each member), and can distribute a fresh SAK when a member is removed from the live membership. If the Key Server has not generated a fresh SAK since the last addition to the Key Server's Live Peer List, an SAK is not distributed in transmitted MKPDUs. A fresh SAK is not distributed until:

a) The Key Server's Live Peer List contains at least one peer, and

b) An MKA suspension is not in progress, i.e., the Key Server either does not support suspension (5.11.4), or the Key Server's suspendedWhile timer is zero (9.18), and

c) Either

   1) MKA Life Time (Table 9-3) has elapsed since the prior SAK was first distributed,
      or

   2) The Key Server's Potential Peer List is empty.

Once a Key Server has generated an SAK, it shall be distributed in each MKPDU transmitted by its principal actor until all live peers that can use the selected Cipher Suite and Cipher Suite capability (9.6.1) report having installed the SAK for receive or until a change in the live membership of the CA requires the generation of a fresh SAK.

NOTE 2—The CP state machine (Figure 12-2) ensures that SAKs are not continually changed faster than they can be installed and used by each of the CA members.

An MKA participant does not retain any record of SAKs used prior to initialization or re-initialization, and uses a fresh MI whenever it is initialized, thus forcing distribution of a fresh SAK whenever it has no record of prior SAK use. An MKA participant is re-initialized, or deleted and a fresh participant created, if the associated SCI is changed. An MKA participant accepts only SAKs distributed by Key Servers that are mutually live, i.e., shall not accept an SAK distributed in any MKPDU that does not contain that participant's MI and acceptably recent MN in the Live Peer List.

NOTE 3—Inclusion in the Potential Peer List is sufficient to prove that the Key Server is live, but not that it has recognized the participant as live and updated the distributed key. Reinitializing and using a new MI if an SCI is changed prevents use of the new SCI in conjunction with a previously distributed SAK, allows the Key Server to check for SCI collision before distributing a fresh SAK, and means that the Key Server and other participants are aware of the change when calculating SSCI values.

Each participant shall record the values of KI, nextPN for secure frame generation (see IEEE Std 802.1AE), for the last SAK accepted from each Key Server (as identified by the MI that forms part of the KI), to ensure that the SAK-nonce pair is not reused if the member switches (or is induced to switch) to a different Key Server and back to a previously used Key Server. Each member shall be capable of recording at least four such KI, nextPN tuples, and shall reinitialize (thus choosing a new MI, and forcing distribution of a fresh SAK) if the number of Key Servers used since the MI was chosen exceeds its ability to maintain all the relevant tuples, irrespective of the time elapsed or perceived to have elapsed since the SAK distributed by a given Key Server was last used.

Once a Key Server has distributed an SAK, it shall not distribute any previously generated SAK in any subsequent MKPDU. However, it is recognized that successive SAKs will have the same value with a probability of no less than 1 in $2^{-keysize}$ when generated as specified (see 9.8.1).

A Key Server shall distribute a fresh SAK whenever it selects a new MACsec Cipher Suite (9.7) unless the new Cipher Suite is the Null Cipher Suite, i.e., plain text transmission is selected. The CP state machine (12.4) ensures that this requirement is met.

9.8.1 SAK generation

Each SAK should be generated using the KDF specified in 6.2.1 using the following transform:

SAK = KDF(Key, Label, KS-nonce | MI-value list | KN, SAKlength)

where

Key	=	CAK
Label	=	“IEEE8021 SAK”
KS-nonce	=	a nonce of the same size as the required SAK, obtained from an RNG each time an SAK is generated.
MI-value list	=	a concatenation of MI values (in no particular order) from all live participants
KN	=	four octets, the Key Number assigned by the Key Server as part of the KI
SAKlength	=	two octets representing an integer value (128 for a 128 bit SAK, 256 for a 256 bit SAK) with the most significant octet first.

‘|’ denotes concatenation of octet strings.

NOTE 1—The CAK and KS-nonce are both secret values that contribute cryptographic entropy to the SAK. While the randomly generated MI-values and the KN are not secret, their addition allows further permutation of the keying material, contributing freshness to the key generation process. Including the KN ensures that this contribution will not repeat unless the Key Server repeats its MI choice, while all other participants repeat or maintain their MI choices. In many deployment scenarios a restart will change MAC_Operational for two connected participants, causing both to reselect their MI.

NOTE 2—Specifying an MI-value list order would be of no value, as key generation is local to the Key Server.

Alternatively, SAKs may be generated directly by the Key Server’s RNG (9.2.1).

9.8.2 Use of AES Key Wrap

Each distributed SAK shall be protected by AES Key Wrap, as specified by IETF RFC 3394. The AES Key Wrap default IV defined in that specification shall be used.

9.8.3 MKPDU application data

Each participant that considers itself to be the current Key Server can distribute an SAK by encoding the following information in transmitted MKPDUs:

a) Distributed SAK, the SAK, protected by AES Key Wrap (see 6.2.4, 9.8.2, Figure 11-11).
b) The KN, 32 bits.

Only one Distributed Key is present in an MKPDU, although each participant can be using up to two SAKs to protect MACsec data (9.10).

9.9 SA assignment

The Key Server identifies the association number (AN) to be used with each key it distributes, using ANs in sequence (0,1,2,3,0,…) unless it yields to a higher priority key server for a period, and beginning the sequence with the first AN following the last SAK in use by any live CA member when the SAK was first distributed.

9.9.1 MKPDU application data

The Key Server encodes the following information in every MKPDU transmitted that includes a Distributed SAK, i.e., that has the Key Present flag set:

a) Distributed AN, an enumeration [0,1,2,3]

9.10 SAK installation and use

Each CA member's KaY uses the LMI supported by the SecY (10.7 of IEEE Std 802.1AE-2018) to create a receive SC for each of its live peers' SCs and a SA for each receive SC and its own transmit SC(s) using the distributed SAK and AN.

NOTE 1—As specified in IEEE Std 802.1AE, an MKA participant is associated with each transmit SC even if multiple transmit SCs (each with its own SCI) are associated with one KaY. Only one of those participants can act as the Key Server at a time. MIs for the others are included in its Live Peer List.

Each participant advertises the status of the receive SAs and its transmit SA. The Key Server will enable transmission for its transmit SA, immediately if it was not previously transmitting or receiving, and when all live peers report that they can receive using the corresponding SAK and AN otherwise. The other participants enable transmission when they see that the Key Server is transmitting using the SAK and AN. See the CP state machine (12.4, Figure 12-2).

If the KaY comprises more than one actor, only SAKs that are received by the principal actor (or distributed by that actor if it is the Key Server itself) are installed.

Each CA member advertises the lowest PN used for each SAK (LLPN, OLPN—Lowest Acceptable PN for the Old Key) within one second prior to transmitting each MKPDU window, and may use the time bounds provided by received LPNs and the reflection of its own MI.MN tuples in the Live or Potential Peer List to discard delayed traffic.

NOTE 2—Enforcement of bounded received delay necessitates transmission of MKPDUs at frequent (0.5 s) intervals, to meet a maximum data delay of 2 s while minimizing connectivity interruption due to the possibility of lost or delayed MKPDUs. MKPDU can therefore operate without data delay protection, to lessen its processing requirements.

When MKA is used in conjunction with an XPN Cipher Suite, an SSCI is required for each SA. As specified in IEEE Std 802.1AE, the SA with the numerically greatest SCI uses the SSCI value 0x00000001, that with the next to the greatest SCI uses the SSCI value 0x00000002, and so on. The value 0x00000000 is not used.

PAEs that implement MKA Version 3 (or higher) order the MIs in the Live Peer List of transmitted MKPDUs. The MI associated with the numerically greatest SCI occurs first in the list, that with the next to greatest SCI second, and so on. The Key Server's own MI is not included in the Live Peer List. The SSID of the Key Server is also encoded in each MKPDU used to distribute an SAK for an XPN Cipher Suite, and indicates not only the value to be used by the Key Server, but also is corresponding to the position that it would otherwise occupy in the list. This information allows a receiving participant to determine the SCI to SSCI mappings for the transmit SAs used by it and each of the participants in the MKPDU's Live Peer List that are also in its own Live Peer List, even if those lists differ as a result of MKPDU loss or delay.

On receipt of a Version 3 or higher MKPDU distributing an SAK for an XPN Cipher Suite, a PAE implementing MKA Version 3 determines SSCI values as follows. The Key Server's SA takes the SSCI value explicitly encoded in the Live Peer List in the MKPDU. The SSCIs are then assigned in order to each of the SAs. The MI values that have a position ordinally lower than the Key Server's SSCI are given their ordinal value. The MI values that have a position in the Live Peer List of the Key Server's SSCI or greater are given an SSCI of their ordinal value + 1. So, for example, if the Key Server's transmit SA's SSCI is 0x00000002 and there are three MI values in the MKPDU Live Peer List, then their transmit SA's SSCIs are 0x00000001, 0x00000003, and 0x00000004, respectively. A receive SA is not created for any MI that is not in the Live List of a participant receiving a distributed SAK, but the ordered Live List from the MKPDU distributing that SAK is retained so the receive SA can be created when an MKPDU with that MI, proving liveness and conveying an SCI, is received.

NOTE 3—Since SSCIs are assigned sequentially and MKPDUs can convey information for only a limited number of participants, only the least significant octet of the Key Server's SSCI is encoded in MKPDUs; each of the more significant octets has the value 0x00.

9.10.1 MKPDU application data

Each CA member encodes the following information in every MKPDU transmitted, for the latest and the old AN:

a) AN

b) KI, the Key Identifier for an SAK

c) LPN, Lowest Acceptable PN (least significant 32 bits for XPN Cipher Suites)

d) Receiving, TRUE if reception for the SAK is enabled for all the receive SAs identified by AN

e) Transmitting, TRUE if the SecY is transmitting using the SAK and its own SA with that AN

f) Delay Protect, TRUE if LPNs are being reported sufficiently frequently to allow the recipient to provide data delay protection. If FALSE, the LPN can be reported as zero

A fixed format encoding is used. For convenience, these fields can be identified by the names and acronyms Latest AN, Old AN (LAN, OAN), Latest Key Identifier/Old Key Identifier (LKI/OKI), Lowest Acceptable PN for the Latest Key/Lowest Acceptable PN for the Old Key (LLPN/OLPN), Latest Receiving/Old Receiving (LRX/ORX), Latest Transmitting/Old Transmitting (LTX/OTX).

NOTE 1—The Latest and Old SAKs were not necessarily distributed by the same Key Server, or by the current Key Server. Both can be receiving at the same time, to enable transition from one SAK to the next without frame loss, although only one will be transmitting at any instant.

NOTE 2—When an XPN Cipher Suite is used the most significant 32 bits of the Lowest Acceptable PNs for both ANs is encoded as specified in 9.18.7.

PAEs that implement MKA Version 3 (or higher) order the MIs in the Live Peer List of transmitted MKPDUs as specified above (9.10) and encode the Key Server's transmit SA's SSCI in every MKPDU used to distribute an SAK for an XPN Cipher Suite.

9.11 Connectivity change detection

Changes in CA membership represent changes in the topology of a network that would be accompanied by link level indications, if the connectivity association represented by the CA were a LAN supported directly by media access method specific procedures, modeled by changes in the MAC_Operational and OperPointToPointMAC status parameters (IEEE Std 802.1AC, IEEE Std 802.1Q). The SecY that operates MACsec detects some of the conditions that cause such topology changes, and notifies the client of its Controlled Port through changes in the status parameters. Some of the changes that the SecY cannot detect are unimportant, and require no exceptional measures, while other information about CA membership can be accessed through the LMI to optimize the operation of certain client protocols—loss of connectivity to a Designated Router or Designated Bridge (for example) might be detected by MACsec more rapidly than by those protocols. A more significant change is the creation of new connectivity, which can only be detected by client protocols when it has occurred as opposed to when it is about to occur. New connectivity that is not signaled by MAC_Operational can cause temporary data loops in bridged networks, while a TRUE-FALSE-TRUE transition in MAC_Operational has the defined effect (for spanning tree protocols) of re-initializing state machines to deal with new connectivity.

9.12 CA formation and group CAK distribution

If the Key Server so decides, a group or pairwise CAK, can be used to distribute a group CAK and a corresponding CKN to each member of a potential group CA. Where the Key Server is the network access point and is being attached to by a number of desktop devices, the latter need have no a priori knowledge of whether they are going to be in a group CA or not. The reference table for Key Server Priority recommends a range for network access points, so they and not the desktop devices are preferentially selected as Key Server(s). The Key Server's KaY should not delete an MKA participant for a CAK that is used to distribute a further CAK until that further CAK has been used to establish peer liveness with each of the other CA members that are live peers of that distributing participant. Similarly each of the other CA members should not delete a participant used to receive a CAK until peer liveness with the Key Server has been established using the received CAK. Participants used to distribute and receive CAKs are not necessarily deleted once a CAK has been successfully deleted, but can remain active for future use.

9.12.1 Use of AES Key Wrap

Each distributed CAK shall be protected by AES Key Wrap, as specified by IETF RFC 3394. The AES Key Wrap default IV defined in that specification shall be used.

9.12.2 MKPDU application data

Each participant that considers itself to be the current Key Server and wishes to distribute a CAK encodes the following information in every MKPDU transmitted with a non-null Live Peer List:

a) Distributed CAK, the CAK, protected by AES Key Wrap (see 6.2.4, 9.12.1, Figure 11-13)

b) The CKN, an integral number of octets up to 32

Only one Distributed CAK is present in each MKPDU, and the MKPDU shall not contain a Distributed SAK.

NOTE 1—The Key Server will terminate the MKA Instance after a sufficient number of transmissions have ensured that the intended recipient will either have received the CAK or have been removed from the Live Peer List.

NOTE 2—The AES Key Wrap does not include the CKN (up to 32 octets) since there is no security exposure if the two are detached—the worst that can happen is that the CKN fails to locate the key.

9.13 Secure announcements

EAPOL-Announcement PDUs are not transmitted securely, but can be used to make decisions (e.g., NID selection) prior to authentication. Announcements can also be transported by MKA so that the data used to make those decisions can be retrospectively confirmed and so the participant's PAE can choose to use other authentication selections if appropriate.

9.13.1 MKPDU application data

Each participant that sends EAPOL-Announcements shall include the following Announcement parameters (if available) in each MKPDU within an MKA Announcement parameter set (Figure 11-15):

a) The NID Set associated with the CAK for this MKA instance.

b) The NID Set for the NID currently marked as Access requested (Table 11-9), if different from that associated with the CAK for this MKA instance.

Each such participant should also include the following parameters (if available), in priority order, subject to sufficient space remaining in the MKPDU for encoding the other parameters required by this specification:

c) Global Announcement parameters, i.e., those not within the context of any NID Set.

d) Other NID Sets.

NOTE—In contexts where a NID might be used, Global Announcement parameters can be referenced by using a null, i.e., zero length, NID name. Repeating those parameters in an announcement with a NID set with an explicit zero length name is not prohibited, but serves no useful purpose. The term 'null NID' thus refers to both Global Announcement parameters and additional parameters provided in such a NID set.

9.14 MKA participant creation and deletion

The creation of MKA protocol participants is initiated by the PAE's Logon Process (12.5) or an equivalent system process. The Key Agreement Entity (KaY) creates participants on request, and as a possible consequence of its own operation, and coordinates their operation. Each KaY shall be capable of maintaining the simultaneous operation of at least two participants for the permanently present Common Port and at least two for each virtual port.

An MKA participant shall be created when a CAK, CKN tuple (see 9.3) becomes available as a result of one of the following:

a) Administrative configuration of a pre-shared key (PSK, 6.3.3).

b) Successful authentication using EAP (Clause 8, 6.2.2).

c) Receipt of a CAK, distributed by a key server, by an MKA participant (9.12).

An MKA participant can also be created, using previously cached or explicitly configured (PSK) CAK, CKN tuples as a result of the following:

d) The SecY's or PAC's Common Port becomes operational and the Logon Process wishes to reactivate a recently active participant.

e) Receipt of an MKPDU with a CKN that matches that for a cached CAK.

f) Receipt of an announcement containing a desired NID, with a KMD that matches that for a cached CAK for the NID.

g) The participant is explicitly activated by management.

An MKA participant shall be deleted as a result of any of the following:

h) The CAK lifetime (if specified) has expired.

i) The CAK was derived from an EAP exchange, but has not resulted in the recognition of a Live Peer (9.4) with an acceptable MACsec Capability (Table 11-6) within a period MKA Life Time (Table 9-3).

An MKA participant may be deleted as a result of any of the following:

j) The last key server to distribute a key using that CAK is no longer in the participant's Live Peer List (9.4.3, 9.4.4) but is (as identified by its SCI) on the Live Peer List of another participant that is using the same Common Port.

k) The number of participants would otherwise exceed the number that can be supported by the system.

9.15 MKA participant timer values

Table 9-3 summarizes each MKA participant's use of timers and their timeout values.

Table 9-3—MKA Participant timer values

Timer use	Timeout (parameter)	Timeout (seconds)
Per participant periodic transmission, initialized on each transmission, transmission on expiry (9.4).	MKA Hello Time or MKA Bounded Hello Time	2.0 0.5
Per peer lifetime, initialized when adding to or refreshing the Potential Peer List or Live Peer List, expiry cause removal from the list (9.4.3).	MKA Life Time	6.0
Participant lifetime, initialized when participant created or following receipt of an MKPDU, expiry causes participant to be deleted (9.14).		
Delay after last distributing an SAK, before the Key Server will distribute a fresh SAK following a change in the Live Peer List while the Potential Peer List is still not empty.		
Maximum suspendFor value. The maximum suspendedWhile value is MKA Life Time longer.	MKA Suspension Limit	120.0

9.16 MKA management

The PAE management process controls and monitors the operation of the KaY and MKA participants, providing access for network management through the LMI. The following variables (see also 12.2) can be used to manage the operation of the KaY for a given port (as identified by its portNumber, see 12.9):

- enable: Set by management to enable (clear to disable) the use of MKA.
- active: Set if there is at least one active actor, transmitting MKPDUs.
- authenticated: Set if the principal actor has determined that Controlled Port communication should proceed without MACsec.
- secured: Set if the principal actor has determined that communication should use MACsec.
- failed: Cleared when authenticated or secured are set, set if the latter are clear and MKA Life Time (Table 9-3) has elapsed since an MKA participant was last created.
- actorSCI: The SCI assigned by the system to the port (applies to all the port's actors).
- actorPriority: The Key Server Priority for all the port's actors.
- keyServerSCI: The SCI for Key Server for the principal actor. Null if there is no principal actor, or that actor has no live peers. Matches the actorSCI if the actor is the Key Server.
- keyServerPriority: The Key Server Priority for the Key Server for the principal actor. Matches the actorPriority if the actor is the Key Server.
- joinGroup: Set if the KaY will accept Group CAKs distributed by MKA.
- formGroup: Set if the KaY will attempt to use point-to-point CAKs to distribute a Group CAK, if it is the Key Server for the MKA instances for all the point-to-point CAKs.
- newGroup: Set by management if a new Group CAK is to be distributed, if the KaY is the Key Server for the MKA instances for all point-to-point CAKs. Cleared by the KaY when distribution is complete.
- macsecCapable: Set, for the port and applicable to all actors, by management. See 9.6.1.
- macsecDesired: Set, for the port and applicable to all actors, by management. See 9.6.1.
- macsecProtect: As used by the CP state machine, see 12.4.
- macsecReplayProtect: As used by the CP state machine, see 12.4.
- macsecValidate: As used by the CP state machine, see 12.4.
- macsecConfidentiality: Set by management, the confidentiality offset (9.7.1) advertised to all actors when acting as a Key Server.
- txKN: The Key Number (KN, 9.8) assigned by the Key Server to the SAK currently being used for transmission. Null if MACsec is not being used.
- txAN: The AN (9.9) assigned by the Key Server for use with txKN. Zero if MACsec is not in use.
- rxKN: The Key Number assigned by the Key Server to the oldest SAK currently being used for reception. The same as txKN if a single SAK is currently in use (as will most often be the case). Null if MACsec is not being used.
- rxAN: The AN (9.9) assigned by the Key Server for use with rxKN. The same as txAN if a single SAK is currently in use. Zero if MACsec is not in use.
- suspendFor: Set by management to a non-zero number of seconds between 1 and MKA Suspension Limit to initiate a suspension (9.18) of that duration (if the KaY's principal actor is the Key Server) or to request a suspension (otherwise).

— suspendOnRequest: Set by management to allow the KaY's principal actor to initiate a suspension if it is the Key Server and another participant has requested a suspension.

— suspendedWhile: Read by management to determine if a suspension is in progress and (when available) to discover the remaining duration of that suspension.

NOTE 1—Information that can be obtained from the SecY MIB is not duplicated in the above variables.

The KaY maintains the following variables for each active and potential participant:

— CKN: The CKN for the participant.

— KMD: The KMD for the participant.

— NID: The NID for the participant.

— authData: Authorization data associated with the CAK. See 6.3.

— cached: Set by the KaY if the participant's parameters are cached. If set, cached can be cleared by management to remove the participant from the cache.

The participant's CAK shall not be readable by management.

NOTE 2—Specifying authorization data is beyond the scope of this standard. Use of Radius attributes to configure Controlled Port clients is typical. See Annex D.

The following per participant variables can be used by management to monitor and control activation:

— active: Set if the participant is active, i.e., is currently transmitting periodic MKPDUs.

— retain: Set by management to retain the participant in the cache, even if the KaY would normally remove it (due to lack of use for example).

— activate {Default, Disabled, OnOperUp, Always}: Controls when the participant is activated. Cached entries created by the KaY as part of normal operation, without explicit management, have the value Default, and are activated according to the implementation dependent policies of the KaY (see 9.14). This variable can be set to any of its values by management. Disabled allows the cache entry to be retained, but disabled for an indefinite period. OnOperUp causes the participant to be activated when the PAE's Uncontrolled Port becomes MAC_Operational and when the PAE resumes following suspension (9.18). Always causes the participant to remain active all the time, even in the continued absence of partners. If the value is changed to Disabled the participant ceases operation (is deleted) immediately.

PSKs (potential MKA participants) may be created in the CAK cache by management, specifying CKN, CAK, KMD, NID, and authorization data (as required). Entries are initially created with activate Disabled and retain set. An implementation shall provide a way of disabling management creation of participants, as it offers an avenue of attack, through creation of participants with CAKs known to the attacker.

The KaY maintains the following variables for each active participant:

— principal: Set if the participant is currently the principal actor.

— livePeers: A list of the SCI's of the participant's live peers (9.4).

— potentialPeers: A list of the SCI's of the participant's potential peers (9.4).

— distCKN: The CKN for the last CAK distributed (either by the actor or one of its partners). Null if this participant has not been used to distribute a CAK.

9.17 MKA SAK distribution examples

This clause (9.17) provides some examples of MKA operation, focusing on the essentials of Key Server election and MACsec SAK distribution, installation, and use. The parts of the MKA transport component of each MKPDU—the Actor's member identifier and message number, and the member identifier and message number of each participant in the Live Peer List or Potential Peer List—and the other parameters transmitted are shown as follows:

Actor : Live Peer List : Potential Peer List: other parameters

where each MI, MN tuple is shown as X+1, X+2, etc. and tuples in the peer lists are separated by commas.

9.17.1 Two participants

Consider two stations S_A, S_B each with an MI,MN of A+.., B+... Assuming that S_A has the higher key server priority and chances to transmit first following the initialization of both participants (probably as a result of a point-to-point LAN becoming operational), the message exchange proceeds as follows:

S_A → A+1 : : : → S_B.. (1.1)

S_B → B+1 : : A+1 : → S_A.. (1.2)

S_A now believes that S_B is live, and will distribute an AES wrapped SAK with identifier A+1, assigning it to AN 0, and enabling S_A's receiver and transmitter:

S_A→ A+2 : B+1 : : DistributeSAK = {SAK}A+1; SAKuse = A+1.0.rt → S_B.. (1.3)

Note that S_A is able to enable both receiver and transmitter because it was not communicating previously, so nothing is lost by adopting the new key immediately. On receipt of this message S_B will believe that S_A is live, and will install the distributed SAK, and start transmitting and receiving

S_B → B+2 : A+2 : : SAKuse = A+1.0.rt → S_A.. (1.4)

If S_B, with the lower key server priority, transmits first, the exchange proceeds as follows:

S_B → B+1 : : : → S_A.. (2.1)

S_A → A+1 : : B+1 : → S_B.. (2.2)

S_B → B+2 : A+1 : : → S_A.. (2.3)

S_A → A+2 : B+2 : : DistributeSAK = {SAK}A+1; SAKuse = A+1.0.rt → S_B.. (2.4)

On receipt of this fourth message, S_B can also install the key and start transmitting and receiving.

9.17.2 Another participant joins

Continuing with the prior example (9.17.1), a station S_C is attached to the LAN or powered up. Assuming that S_C is just in time to receive the last message:

S_A → A+2 : B+2: : DistribSAK = {SAK}A+1; SAKuse = A+1.0.rt → S_B, S_C .. (3.4)

Then S_C will record both S_A and S_B as potential peers and send:

S_C → C+1: : A+2, B+2: → S_A, S_B .. (3.5)

This message will prove S_C's liveness to both S_A and S_B. S_A, acting as the Key Server, will distribute a fresh key, assigning it to AN 1 and enabling its receiver:

S_A → A+3 : B+2, C+1 : : DistribSAK = {SAK}A+2; SAKuse = A+1.0.rt, A+2.1.r → S_B, S_C .. (3.6)

On receiving this message S_B can install the new key, and start to receive, and can send:

S_B → B+3 : A+3, C+1 : : SAKuse = A+1.0.rt, A+2.1.r → S_A, S_C .. (3.7)

S_C also installs the new key, and starts to receive and transmit, and will send:

$S_C \rightarrow$ C+2 : A+3, B+3 : : SAKuse = A+2.1.rt $\rightarrow S_A, S_B$ .. (3.8)

Following receipt of 3.6 and 3.8 S_B can start to transmit as well as receive using the new key, as can S_A following receipt of 3.7 and 3.8. As both S_A and S_B were previously transmitting they need to know that all active participants can receive using the new key, before they use it to transmit.

9.18 In-service upgrades

The control plane software of a system that is a member of a CA can be upgraded, temporarily suspending MKA operation, without interrupting the secure data connectivity provided by the CA, if the system and each of its peers in the CA support in-service upgrades (5.11.4). An SAK that is already in use will continue to be used provided that each of these peers does not conclude that it is the CA's only active member, and provided that a fresh SAK is not distributed by a recognized Key Server. If each member's suspendedWhile timer is not zero these conditions will be met (12.2, 12.4, 12.5).

NOTE—A KaY's suspendedWhile timer takes a non-zero value to indicate a suspension, a period during which one or more of its principal actor's partners has ceased (or can be expected to cease) transmitting MKPDUs.

This standard specifies procedures that allow a Key Server to control the initiation, proposed duration, and possible early termination of any MKA suspension by communicating the current value of the suspendedWhile timer in MKPDUs. A Key Server that supports in-service upgrades always includes the value in each MKPDU transmitted. A suspension is terminated by expiry of a participant's suspendedWhile timer, by the Key Server communicating a zero value, or by a Key Server that does not support in-service upgrades omitting the value from transmitted MKPDUs.

9.18.1 Initiating suspension

A participant other than the Key Server can request a suspension by transmitting a (non-zero) suspendFor value in an MKPDU. The participant can maintain the request for an indefinite period by repeating the value in each MKPDU transmitted before it suspends MKA operation. If the request is no longer relevant, perhaps because the participant has upgraded its control plane software without waiting for a suspension, the participant can drop the request by transmitting MKPDUs with a suspendFor value of zero.

The Key Server records, for each of the participants on its Live Peer List, the lesser of the values of the suspendFor parameter in the last MKPDU received and the MKA Suspension Limit specified in Table 9-3.

A Key Server should not initiate a suspension until it has started transmitting using the last SAK it has distributed and is no longer receiving using any prior SAK. This requirement is equivalent to stating that the CP state machine should be in state RETIRE having completed the actions on entry to that state (see Figure 12-2). If these conditions are satisfied, and the value of the Key Server's own suspendFor parameter is non-zero or the Key Server's policy control suspendOnRequest is True and one of the received suspendFor parameters is non-zero, the Key Server will initiate a suspension. The Key Server can apply additional policy controls on the setting of its suspendFor and suspendOnRequest parameters to limit when a suspension can occur, or to limit the frequency of suspensions. Any participant can also apply its own policy controls, limiting (for example) how long it is prepared to wait for the Key Server to initiate a suspension of adequate duration.

To initiate a suspension the Key Server sets the value of its suspendedWhile timer to the greatest of the applicable suspendFor values. Subsequently suspendedWhile is decremented once per second, and then set to the greater of its new value and the greatest applicable suspendFor value. The value of suspendedWhile (decrementing over time) is transmitted in all MKPDUs for all MKA instances that have elected it Key Server. These transmissions should persist for at least MKA Life Time before the Key Server suspends its own operation (if desired).

A participant, other than the Key Server, that wishes to suspend its own operation of MKA includes a non-zero suspendFor value in all MKPDUs transmitted and should not suspend its operation of MKA until it receives an MKPDU from the Key Server with a non-zero suspendedWhile value that is greater than or equal to the value of its suspendFor parameter.

NOTE—Details of MKA operation are likely just a part of the concerns to be addressed when performing a system software upgrade, and the overall considerations will not necessarily allow the KaY to delay suspension to facilitate transmission and reception of MKPDUs as recommended. The probability that the transmission of a single MKPDU by the participant requesting suspension will be successfully received and acted on by the Key Server might be acceptable, as might the probability of reception of a single MKPDU transmitted by the Key Server.

If the requesting participant does not suspend its own operation of MKA, but continues to transmit a non-zero suspendFor value in subsequent MKPDUs, then the Key Server's suspendedWhile value will continue to be at least that of the requested value—provided that the Key Server does not suspend its own operation and its suspendOnRequest parameter remains True. The MKA Suspension Limit specified in Table 9-3 limits the time for which a participant will maintain existing connectivity after all partners requesting suspension have been removed from its Live Peer List, on the assumption that these partners have been suspended and will resume operation.

9.18.2 Suspending

A CA member should not suspend its own operation, unless it is

a) Receiving and transmitting using a single SAK, or
b) Constrained by policy controls that place limits on the time that it is prepared to wait to suspend.

9.18.3 Suspended members

A CA member that suspends MKA operation while using MACsec to secure connectivity continues to generate, transmit, receive, and verify secure data frames as specified by IEEE Std 802.1AE, maintaining the receive and transmit SAs current at the time of the suspension. In effect, the CP state machine remains in state RETIRE.

A suspended member is not required to retain any record of its MI, or of its Live Peer List or Potential Peer List, for any of its participants. It requires a record of any SAK(s) used by transmit and receive SAs only in so far is required by the operation of those SAs, and shall not redistribute or otherwise share the SAK(s) with other participants while suspended or at any later time.

A suspended member does require access, when operation is resumed, to the following information required by the operation of each SA in use:

a) The SCI
b) The AN

A suspended member is not required to retain the KI of any SAK in use, but can report a zero value (after resuming operation) when responding to management requests and completing KI fields in MKPDUs. The rules for SAK generation (9.8) ensure that a fresh SAK will be distributed after a suspended member resumes operation with a new MI.

A member can suspend MKA operation because part of its system's functionality will not be available during an in-service upgrade. The member might not be able to transmit and receive frames because software associated with a particular interface module is to be upgraded, for example. The suspended member can retain its MI provided that it continues to operate MKA, with the exception that any MKPDUs that might be generated for transmission or that are received can be discarded. The normal operation of MKA will result in the removal of partners from the Live Peer List and Potential Peer List (9.4.3) if MKPDU loss persists.

9.18.4 Resuming operation

When a CA member resumes, it sets its own suspendedWhile timer value to Max Suspension Limit or to some lower policy determined limit. It also sets its suspendFor parameter value to zero once it has determined that the upgrade has completed successfully (see 9.18.6).

NOTE 1—The Max Suspension Limit is specified in Table 9-3.

The CA member might have suspended only because it was unable (for the duration of the suspension) to receive or transmit MKPDUs or install fresh SAKs. If it has continued to operate MKA while suspended, its MI, Live Peer List, and Potential Peer List will be retained when it resumes. Otherwise, i.e., if MKA operation ceased, it will select a fresh MI and its Live Peer List and Potential Peer List will be initialized and empty on resumption.

If the Key Server that initiated the suspension suspends itself, then it is possible that a participant for another CA member (that might or might not have suspended itself) will be elected Key Server before the initiating Key Server resumes. If the newly elected Key Server's suspendedWhile timer is running, then it will distribute the (decrementing) value of that timer and thus prolong the suspension. This provision means that it is unnecessary for a resuming member to remember whether it was or was not the Key Server prior to being suspended. If the initiating Key Server resumes operation before the timer expires it will once more assume responsibility for monitoring the suspension.

The current Key Server shall terminate the suspension before the value of its suspendedWhile timer reaches zero by resetting that value (included in all transmitted MKPDUs) to zero under the following conditions:

a) Either
   1) For every active receive SA it has a live peer with an SCI that matches that of the receive SA;
   or
   2) It has a live peer with an SCI that does not match any existing receive SA;
   and
b) It has recorded a value of zero for the suspendFor parameter received from each of these live peers.

To determine, for the purpose of test a)1), that a receive SA is active the Key Server monitors the value of the InPktsOK management counter (see Figure 10-5 and 10.7.9 of IEEE Std 802.1AE–2018) for the SA. The receive SA is not active if the counter has not been incremented for MKA Life Time. Test a)2) detects the addition of a new CA member.

NOTE 2—For the common case of a point-to-point CA, these conditions simplify to either having a live peer communicating a zero suspendFor value or not having received any secured frames for MKA Life Time.

NOTE 3—While the receipt of validated secured data frames does not ensure that communication is with a live peer in current possession of a shared SAK, the lack of any such reception for a period of MKA Life Time (6 seconds) is a strong practical indication of the absence of such a peer.

9.18.5 XPN support

If a receiver loses more than 2^{30} consecutive frames when an XPN capable Cipher Suite is being used, the 32 most significant bits of the PN of the next frame to be received might be recovered incorrectly. If more than 2^{32} frames are lost, these most significant bits will be recovered incorrectly. All subsequent secured frames will fail validation and be discarded, unless some means other than the receipt of secured data frames is used to determine their value.

When a suspension is in progress, the risk of losing a large number of frames is increased as the operation of supervisory protocols that would otherwise detect temporary loss of connectivity might also be suspended.

At 100 Gb/s 2^{30} minimum sized secured frames can be transmitted in 10 seconds, well within the potential duration of a suspension. To ensure that the most significant bits are recovered when all CA members resume MKA operation, the most significant 32 bits of the Lowest Acceptable PN for the Latest Key and the Old Key are communicated when in-service upgrades are supported and any XPN capable Cipher Suite is being used.

9.18.6 Managing in-service upgrades

Careful planning is required when upgrading systems that compose a network if the network is to remain in operation, and if the costs of recovering from a failed upgrade are considerable. Best practices include the following:

a) Ensuring that the network manager has an up to date record of all the systems in the network, and of the network configuration.

b) Ensuring that the network manager has an up to date and independently backed up record of the software, software revision levels, and configuration parameters currently used by the systems to be upgraded, by their neighbors, and by their other peers in the network.

c) Off-line verification and testing for compatibility between the proposed new software and configuration parameters and the existing software and configuration of neighbors and other peers.

d) Retention of the existing software and configuration parameters by the systems being upgraded until successful operation with the new software and parameters is confirmed.

e) Use of a ‘dead man’ timer by the system to be upgraded, so that the system will automatically revert to the prior software and configuration if satisfactory management communication cannot be established with the network operations center after the upgrade.

f) Off-line verification and testing of any such fallback mechanism.

g) Whenever possible, upgrading only one intermediate system (bridge or router) at a time, confirming the success of the upgrade before upgrading additional systems.

In providing continued secure data connectivity while an in-service upgrade is performed, MKA makes a modest contribution to the task of upgrading network systems. MKA lacks the knowledge and scope to ensure or to check that best practices are being followed when the upgrade is being performed, and the network administrator is not relieved of these responsibilities.

The maximum time allowed for suspension, 120 seconds, is believed to be adequate for an upgrade, followed by expiry of the dead man timer, and reversion to the original software revision and configuration. However, if MKA successfully resumes operation, it could begin to distribute a new SAK at a time when the network operation center might not (for reasons completely unrelated to the use of MKA or MACsec) have re-established management connectivity with the upgrading system. If the system is then unilaterally suspended by operation of the dead man timer, it is possible that the members of the CAK will not all converge on use of the latest, or the old, SAK. The Key Server shall not redistribute the previous SAK (9.8). As a consequence, if a dead man timer or similar mechanism is being used by a resuming system, that system should not reset its suspendFor parameter to zero (see 9.18.4, 9.8) until the dead man timer has been reset.

MKA’s in-service upgrade support can be deployed in environments where a comprehensive approach to system upgrade is already in place, and already synchronizes update and suspension activities. Such an existing approach can suspend MKA operation, as required, by coordinating the setting of suspendFor and suspendOnRequest parameters using each system’s LMI. The values of suspendedWhile parameters for both Key Servers and other participants may also be set directly using the LMI, thus avoiding adding additional protocol dependencies to the existing coordination mechanism. Any such directly set values shall be consistent with the values that MKA would, and if not suspended will, communicate.

If the suspendFor or suspendedWhile timer values are set (either by using the MIB specified in Clause 13, or through the operation of other protocols) when there is no need for a suspension (i.e., the conditions for terminating the suspension are already satisfied) the suspension is terminated immediately and the values reset prior to including the timer value in any subsequent MKPDU. The conditions for such an immediate termination could occur as a result of one CA member initiating a suspension after other members have downloaded new software that enables them to upgrade during the suspension but before their own timer values are set.

9.18.7 MKPDU application data

Each participant that is capable of supporting in-service upgrades shall include the following parameter in each MKPDU transmitted (see Figure 11-16):

a) MKA suspension time.

   The value transmitted is that of the suspendedWhile timer if the sending participant considers itself to be the Key Server for the MKA instance (i.e., has set bit 8 in octet 3 of the Basic Parameter Set, see Figure 11-8), and is the value of the suspendFor parameter otherwise.

and, when the Current Cipher Suite uses extended packet numbering, the following parameters:

b) The 32 most significant bits of the Lowest Acceptable PN for the Latest Key.

c) The 32 most significant bits of the Lowest Acceptable PN for the Old Key (if in use).

A receiving participant sets its member's suspendedWhile timer to a received suspendedWhile value iff it is the member's principal actor and agrees that the transmitter is the Key Server for its MKA instance.

A receiving participant records a suspendFor value received from any live partner, superseding any prior suspendFor value received from that partner.

9.19 In-service upgrade examples

This subclause (9.19) provides some examples of MKA operation, focusing on suspension. The parts of the MKA transport component of each MKPDU—the actor's member identifier and message number, and the member identifier and message number of each participant in the Live Peer List or Potential Peer List—and (some of) the other parameters transmitted are shown, as in 9.17, as follows:

Actor : Live Peer List : Potential Peer List: other parameters

where each MI, MN tuple is shown as X+1, X+2, etc. Initial MN values in these examples are arbitrary. Tuples in the peer lists, and other parameters, are separated by semi-colons.

9.19.1 Requested by end station in point-to-point CA

An end station port S, that is not the Key Server, requests suspension from the Key Server, K, by sending an MKPDU with a suspendFor value (encoded in the MKA Suspension Time field) of 60 seconds. K has suspendOnRequest True, and responds with an MKPDU with a suspendedWhile value (again encoded in the MKA Suspension Time field) of 60 seconds. On receipt, S suspends itself.

S_A	→ A+47:F+63::suspendFor = 60	→	K_F.. (1.1)
K_F	→ F+64:A+47::suspendedWhile = 60	→	S_A.. (1.2)
S_A	suspends		.. (1.3)

The next two or three periodic transmissions by K occur while S (as A) is still on K_F's Live Peer List, so the value of suspendedWhile remains at 60 seconds. The value of suspendedWhile in subsequent transmissions is decremented over time, but before it reaches zero (having reached 30 seconds in this

example) S resumes with a new MI R. S has not kept accurate track of the duration of the suspension, and simply assumes a suspendedWhile value of 120 seconds, the maximum that can be requested.

K_F → F+65:A+47::suspendedWhile = 60 → .. (1.4)
K_F → F+66:A+47::suspendedWhile = 60 → .. (1.5)
K_F → F+67:::suspendedWhile = 58 → .. (1.6)
...
K_F → F+91:suspendedWhile = 30: → .. (1.7)
S_R resumes, assuming suspendedWhile = 120 .. (1.8)

S then exchanges MKPDUs with K, and is recognized as a live peer. K terminates the suspension, and distributes a fresh SAK, with the key identifier (in this example) of F+2.

S_R → R+1:::suspendFor = 0 → K_F.. (1.9)
K_F → F+92::R+1:suspendedWhile = 30 → S_R..(1.10)
S_R → R+2:F+92::suspendFor = 0 → K_F.. (1.9)
K_F → F+93:R+2::suspendedWhile = 0; DistribSAK = {SAK}F+2;
SAKuse = F+1.0.rt, F+2.1.r → S_R..(1.10)

S can then install and use the fresh key, exchanging MKPDUs as required (see 9.17 for relevant examples).

9.19.2 Initiated by Key Server in point-to-point CA

A Key Server initiates the suspension by sending an MKPDU with a suspendedWhile value (encoded, as always, in the MKA Suspension Time field) of 60 seconds, before suspending. In this particular example the other CA member takes the opportunity of suspending and upgrading itself at the same time.

K_F → F+64:A+47::suspendedWhile = 60 → S_A.. (2.1)
S_A suspends .. (2.2)
K_F → F+65:A+47::suspendedWhile = 60 → .. (2.3)
K_F → F+66:A+47::suspendedWhile = 60 → .. (2.4)
K_F suspends .. (2.5)

K resumes, assuming an MI of D, and a suspendedWhile value of 120 seconds.

K_D resumes, assuming suspendedWhile = 120 .. (2.6)
K_D → D+1:::suspendedWhile = 120 → .. (2.7)
K_D → D+2:::suspendedWhile = 118 → .. (2.8)

S resumes, assuming an MI of G, and a suspendedWhile value of 120 seconds.

S_G resumes, assuming suspendedWhile = 120 .. (2.9)
S_G → G+1:::suspendFor = 0 → K_D..(2.10)
K_D → D+3::G+1:suspendedWhile = 118 → S_G..(2.11)
S_G → G+2:D+3::suspendFor = 0 → K_D..(2.12)
K_D → D+4:G+2::suspendedWhile = 0; DistribSAK = {SAK}D+1;
SAKuse = 0.0.rt, D+1.1.r → S_G..(2.13)

The installation and use of the fresh key now proceeds as before.

9.19.3 Intermediate systems suspending multiple CAs

An intermediate system, a router or bridge, will usually have to suspend MKA operation in the multiple CAs that it connects, as a control plane software upgrade will affect all of its ports. It might be the Key Server for some CAs and not for others. Careful planning is required (see 9.18.6) when upgrading intermediate systems in a network, and there are limits to the assistance and safeguards that can be provided by the operation of a local protocol, such as MKA. However, MKA, as shown in this deliberately complex example, does provide some help when multiple systems are involved and there has been a lack of coordination.

In this example, an intermediate system I has ports 1, 2, 3 with MKA participants I1, I2, I3, and neighbors A, B, C, respectively. I2 is already participating in a suspension initiated by B, which has suspended itself. The network administrator instructs I to upgrade, causing suspendFor to be set 60 seconds on each port. I1 is the Key Server for its CA, and can set and start transmitting suspendedWhile immediately. However it still has to arrange to suspend operation on its other ports, so it cannot suspend I1 immediately, so suspendFor and (as a consequence) suspendedWhile for I1 will remain at 60 seconds for the time being. I2 is also (in Bs absence) the Key Server for its CA, and can also set suspendedWhile directly. C, rather than I3, is the Key Server for the third port's CA.

$I2_P$ suspendedWhile =15 .. (3.1)

I1 suspendFor = 60, I2 suspendFor = 60, 13 suspendFor = 60 .. (3.2)

$I1_J$ → J+31:V+60::suspendedWhile = 60 → A_V.. (3.3)

$I2_P$ → P+29:R+58::suspendedWhile = 60 → B_R.. (3.4)

$I3_Q$ → Q+33:W+62::suspendFor = 60 → C_W.. (3.5)

I might have to persist with these transmissions for some time, but if (and as soon as) C has suspendOnRequest set it will respond, and this can happen immediately.

C_W → W+63:Q+33::suspendedWhile = 60 → $I3_Q$.. (3.6)

I can then suspend and upgrade. While it is still suspended B might resume, but its newly assumed suspendedWhile value of 120 seconds will provide sufficient time for I2's suspension (B was not operating when I2 suspended so has no information that would allow it to adopt a lower value).

B_K → K+1:::suspendedWhile = 120 → .. (3.7)

When I resumes, the participants for each of its ports and their partners will exchange MKPDUs as usual in order to recognize live peers, and fresh keys will be distributed.

9.19.4 Key Server suspends in a group CA

A, B, and C are members of a group CA. A, the Key Server, is to suspend to allow its control plane to be upgraded, and 30 seconds is believed to be sufficient for this to occur. The process is started by setting A's suspendFor parameter. This causes A to set its suspendedWhile parameter, and to transmit periodic MKPDUs for the next MKA Life Time (6 seconds) before suspending.

A_E → E+14:Y+15,L+14::suspendedWhile = 30 → B_Y, C_L.. (4.1)

For the following MKA Life Time A will remain on B and C's Live Peer Lists, so neither will claim to be the new Key Server. Finally, assuming that B has the higher Key Server Priority, B will become the Key Server and transmit the, by now decremented and continually decrementing, value of suspendedWhile in its periodic transmissions.

B_Y → Y+21:L+20::suspendedWhile = 24 → B_Y, C_L.. (4.1)

When A resumes, it will advertise suspendedWhile as 120 seconds, before exchanging MKPDUs and realizing that the conditions for terminating the suspension have been met. However, if A does not resume for any reason, B will terminate the suspension in 24 seconds.

10. Network announcements

A port-based network access control capable system can be attached to one LAN, then moved and attached to another, possibly connecting to a number of different networks before being reattached to the first. Such system movement or *roaming* is common where the system moved is a host laptop (see 7.1, 7.3, 7.5). Not all network access points that are of interest to a given system's user have the same capabilities, or provide access to the same network or network services, and the system's user may adopt different roles (employee, private individual etc.) at different times and wish to access different networks. Manual selection or pre-selection of the parameters required to establish secure connectivity at each access point can be inconvenient, or impracticable, as can the time taken if the PAE's Logon Process (12.5) were simply to search through all ways of gaining access to a network, for all possible services, in preference order. Receipt of an announcement serves as a valuable hint, and is also useful in port-based network application scenarios (see Clause 7) where a system moves infrequently.

This clause specifies the following:

a) The information announced, and how that information is organized (10.1).
b) How, when, and to whom announcements are made (10.2).
c) How received announcements are used (10.3).
d) The management variables used to control and monitor announcements (10.4).

Clause 11 specifies the format of the EAPOL-Announcements (11.12), EAPOL-Announcement Requests (11.13), MKA Announcement parameters (11.11), and announcement TLVs within EAPOL-Start PDUs (11.6). These formats are extensible and allow definition of Organizationally Specific parameters.

10.1 Announcement information

A potential peer system, newly attached to a network access point, can use received announcements to help answer the following questions:

a) What network(s) or network service(s) are available?

and for each network:

b) Is access to that network already available, or is authentication or secured connectivity required?
c) What authentication and secure connectivity mechanisms are required or available?
d) What credentials should be presented if EAP is to be used?
e) What cached CAKs can be used with a reasonable chance of success?
f) What level of access (authorization) is or may be provided?

This information can influence whether network access is even attempted. The network services of interest can be unavailable, or the security deemed inadequate by the Logon Process' controls (12.5).

The term *network* has a broad meaning in the context of announcements. Access can be granted, and authorization determined, by a AAA server with the objective of providing connectivity, from and through a number of distinct physical networks, to the *network* of a particular organization or to the internet. While the port-based network access control authentication and security mechanisms are limited to those supported by the access point, their use can be influenced by policies for that target *network* or any of the physical networks traversed. Equally the *network* can be accessed by configuring a PVID at the access controlled port, so that a bridging access point forwards frames to a particular VLAN—with some VLANs providing access to more sensitive resources than others. A number of different VLANs can provide paths to the same physical network with differing levels of authorization provided by different access controls on each path: whether those differences represent different 'networks' or not is a matter of administrative convenience.

For the purposes of announcements each network is identified by a NID, a UTF-8 string used by network attached systems to select a network profile, i.e., information for use when accessing that network, such as appropriate EAP credentials. Network profile information can be pre-configured, or can be acquired by storing announcement parameters after the network access experience has been found to be satisfactory.

A given announcement can supply information for several NIDs, specifying the authentication and protection capabilities and requirements for access to each network. Each announcement includes current connectivity and access status information for each NID, either for all systems attached to the LAN or for a specific system, thus indicating what immediate actions can or should be taken by a system to gain access. Clause 11 specifies additional detail and encodings for the following per NID information:

g) Access Status—a coarse indication of the transmitter's Controlled Port MAC_Operational status (IEEE Std 802.1AC) and current level of access resulting from authentication and the consequent authorization controls applied by that port's clients. One of the following:
   1) No Access—other than to authentication services, and to services announced as available in the absence of authentication (see Unauthenticated Access below).
   2) Remedial Access—the access granted is severely limited, possibly to remedial services.
   3) Restricted Access—the Controlled Port is MAC_Operational, but restrictions have been applied by the network that can limit access to some resources.
   4) Expected Access—the Controlled Port is MAC_Operational, and access provided is as expected for successful authentication and authorization for the NID.

h) Access Requested—authenticated access has been requested for this particular NID.

i) Unauthenticated Access—one of the following:
   1) No Access—other than to authentication services (see Access Capabilities below).
   2) Fallback Access—limited access can be provided after authentication failure.
   3) Limited Access—immediate limited access is available without authentication.
   4) Open Access—immediate access is available without authentication.

j) Virtual Port Access—set if access can be provided by a virtual port.

k) Group Access—set if access, when provided, can be as a member of a group CA. When clear, indicates that access will be through pairwise connectivity.

l) Access Capabilities—authentication and protection capabilities supported for the NID:
   1) EAP
   2) EAP + MKA
   3) EAP + MKA + MACsec
   4) MKA
   5) MKA +MACsec
   6) Higher Layer (WebAuth)
   7) Higher Layer Fallback (WebAuth)
   8) Vendor-specific authentication mechanisms

m) Supported Ciphersuites

n) Key Management Domain for the NID

The Access Status and Access Requested information is dynamic, whereas Unauthenticated Access, Access Capabilities, Virtual Port Access, Group Access, supported Ciphersuites, and the KMD do not change as a direct result of authentication attempts. The Access Capabilities, Virtual Port Access, Group Access, and the supported Ciphersuites can change as a indirect result of authentication if the consumption of resources by an authenticated session limits the ability of the announcer to support them. More than one NID can be announced as currently providing access (i.e., as having an Access Status of other than No Access), as NIDs can be aliases, provided that the connectivity provided for those NIDs is substantially the

same. Restricted Access (rather than Expected Access) is usually a result of the network not being able to provide the access expected for the NID. Remedial Access is usually an indication of problems with authentication or authorization. Remedial Access can be provided by the transmitter's Controlled Port, or by a dedicated client using or providing selective relay to or from its Uncontrolled Port (see Connectivity to unauthenticated systems 7.1.3, 7.3.3, 7.5.3). An Access Status of No Access simply means that access is not being provided as a result of authentication or an authentication attempt: in that case Unauthenticated Access (see below) can indicate current connectivity.

The connectivity announced in Access Status can lag successful authentication, as can changes to authorization. Access Requested is an indication that the NID has been supplied in an EAPOL-Start or otherwise determined by the access point, and can be used by the accessing system as a confirmation that the desired authentication is in progress. Access Requested is clear once authentication and authorization are complete. Thus Access Requested allows the accessing system to distinguish between an Access Status of No Access as a temporary condition, awaiting the results of authentication, and an authentication failure. An Access Status of other than No Access, announced for any NID, takes priority over the static Unauthenticated Access information announced for that or any other NID. A change in Access Status can be used as a hint that connectivity has changed, and signaled to the Controlled Port's clients by blipping MAC_Operational to cause client state machines to restart (to acquire a new network address, for example).

NOTE 1—A port with port-based network access control disabled (or not supported) can advertise Unauthenticated Access as Open Access, with Access Status reflecting MAC_Operational.

Access Requested can be set for more than one NID, but only if they are simple aliases of one another. Access Requested can be set for a NID other than that currently providing authenticated access, as existing connectivity can be preserved until authentication for another NID succeeds.

If authentication has succeeded and MKA is used, MKA (and the SecY, if MACsec is used) provides a secure indication of when the accessing system's Controlled Port is to become MAC_Operational. This indication helps client state machines to start (or restart) at the appropriate time, and avoid significant delays in accessing a network due to the loss of initial protocol packets. When MKA is not used, the Access Status (summarizing the current results of authentication—attempted, on-going or yet to be attempted) and Unauthenticated Access information in unprotected EAPOL-Announcements can be used (once the accessing system is aware that the network access point sends announcements). If Access Status is No Access for all advertised NIDs, but Unauthenticated Access for a NID with Access Requested set is Open or Limited Access, then the Announcement indicates that connectivity is being provided for that NID—and the accessing system can decide to set MAC_Operational for its clients immediately if unauthenticated communication is acceptable (see 12.5). Since unprotected EAPOL-Announcements could be sent by an attacker, receipt of the Unauthenticated Access information does not cause the state machine responsible for Controlled Port connectivity (12.4) to override the configured policies, or to prematurely terminate an in-progress EAP exchange. If Unauthenticated Access is Fallback Access, connectivity will not be provided for the NID until indicated by Access Status (most likely with Remedial Access).

If both Access Status and Unauthenticated Access are No Access (for all NIDs) that does not mean that absolutely no frames can be sent or received from the system receiving the announcement, but rather that the system's Controlled Port should not be made MAC_Operational (thus notifying the port's clients of available connectivity). In particular, WoL (see Annex E) and similar frames can be sent and received. Additionally when authenticated access is supported by a virtual port, the Access Status in generic announcements will always be No Access, and Access Requested cannot ambiguously reflect the NID requested for all the authentication attempts made by different individual systems (and will therefore be clear, for those NIDs, in generic announcements). Both Access Status and Access Requested for each individual system seeking access will be conveyed in specific announcements, and that system will know to acquire that information from specific rather than from generic announcements since Virtual Port Access [see item j) above] will be set for the NID in all announcements.

Higher Layer authentication is typically carried out by WebAuth, or a similar method that requires the accessing systems Controlled Port to become MAC_Operational and run network and application layer protocols. If the Access Capabilities include Higher Layer, then access to WebAuth will be provided whatever the values of Access Status or Unauthenticated Access. If the Access Capabilities announced include Higher Layer Fallback, and not Higher Layer, then access to WebAuth will be available after a failed authentication attempt and indicated by an Access Status of Remedial Access. If higher layer authentication and access control are remote, the announcer will not be able to track the subsequent progress of higher layer authentication and no further changes to the announced values of Access Status and Access Requested can be expected (unless a further authentication attempt is made, or another NID selected): the higher layer authentication method is expected to provide its own detailed feedback to the accessing system.

NOTE 2—Modeling detailed behavior of systems using higher layer authentication is beyond the scope of this standard.

Not all of the Access Capabilities announced are necessarily equally acceptable to the recipient of the announcement, even if implemented. Some choices can be rejected by the Logon Process (12.5) controls. These controls may be configured or configurable per NID according to the sensitivity of the tasks to be performed when accessing a given network, or according to requirements imposed by the network administration but not necessarily known to the access point.

10.2 Making and requesting announcements

Announcements can be transmitted, unprotected, in EAPOL-PDUs transmitted by a PAE through the Uncontrolled Port. Announcement parameters are also conveyed in MKPDUs (9.13.1), but parameters other than those for the NID associated with the MKPDU CAK or for the NID selected for an authentication attempt can be omitted if their inclusion would otherwise require omitting other MKPDU parameters as a consequence of frame size restrictions—preference should be given to parameters applicable to the NID.

EAPOL-Announcements may also be transmitted through a Controlled Port, if Controlled Port communication is secured by MACsec. Announcements shall not be transmitted through an unsecured Controlled Port. Other EAPOL-PDU Types shall not be transmitted through a Controlled Port, whether that Controlled Port is secured or not.

NOTE 1—Transmission of protected announcements using MKA has the advantage of making the information available before decisions have to be made about Controlled Port connectivity, and should be preferred to Controlled Port protected EAPOL-Announcements unless the amount of data is large.

NOTE 2—While reception of announcement parameters using MKA can be used to guard against attacks involving addition and or suppression of EAPOL-Announcements by an attacker, implementors are warned against terminating connectivity simply because protected and unprotected announcements do not match exactly. There can be many reasons for differences between the two sets of data. An implementation can check for a more favorable NID in a protected announcement and attempt to establish connectivity to that NID.

MKPDUs are always transmitted with a destination group MAC address (see 11.1), but only participants that possess the MKPDU's CAK will take note of any conveyed announcement parameters. No special mechanism is provided for a participant to solicit announcements, participation in MKA is sufficient.

EAPOL-Announcements can be generic, applicable to all systems attached to the LAN, or specific to a system. Generic announcements are always transmitted using a group destination address (11.1). Specific announcements are addressed to the intended recipient, allowing a system to communicate Access Requested and Access Status (10.1) to each potential user of a virtual port before secure connectivity is established. All PAEs that transmit announcements can transmit generic announcements, and can transmit specific announcements if and only if they support virtual ports (see 6.1, 6.3.6). Each announcement indicates whether it is generic or specific, and if virtual port access is supported for each NID announced.

An access point transmits unprotected generic EAPOL-Announcements when an access controlled Common Port (see 6.1, 6.4) becomes MAC_Operational, then at intervals of Announcement Time (see Table 10-1). On receipt of an unprotected EAPOL-Start or EAPOL-Announcement-Req and whenever Access Requested changes for a NID, an access point that does not support virtual ports will send a generic announcement, while one that does will send a specific announcement with the destination MAC address associated with the virtual port or potential virtual port. This will be the individual source address of the EAPOL-Start or EAPOL-Announcement-Req, where receipt of one of these EAPOL PDUs stimulated the transmission of the announcement.

NOTE 3—If the frame soliciting the announcement conveys a group address in its source address field, it will be discarded, and no announcement sent.

NOTE 4—Each virtual port is supported by MACsec, and hence requires MKA. The periodic transmission of MKPDUs for a CA ensures that a system accessing a network through a virtual port receives regular, protected, specific announcements.

If an EAPOL-Start or EAPOL-Announcement-Request is received with a unrecognized NID, it is processed as if the NID was not present.

An access point that transmits EAPOL-Announcements through a Controlled Port that is secured by MACsec always uses a group address in the destination MAC address field of the EAPOL-PDU.

The transmission of unprotected EAPOL-Announcements shall be subject to a rate limit: no more than twice the specified number of EAPOL-Announcements (see Table 10-1) shall be transmitted through a given Uncontrolled Port to any given destination address in any given interval of one second. A transmitter may apply the limit to the total number of announcements transmitted, or for each destination address separately for specific announcements. If the transmission of an announcement will not exceed the rate limit, it shall be transmitted within the time Announcement Delay specified in Table 10-1. An announcement transmitted as a result of receiving an EAPOL-Start shall be transmitted before any other frame (including EAPOL-EAP frames resulting from starting or restarting EAP) solicited by reception of that EAPOL-Start. A system that has transmitted an EAPOL-Start requesting an announcement shall not wait for that announcement before proceeding with EAP Authentication. However the system can retransmit the EAPOL-Start, restarting authentication, on the first reception of an announcement that includes a previously requested NID without marking that as Access Requested.

NOTE 5—It is imperative that deploying security does not degrade the network users' experience. An accessing system does not wait for an announcement because it is likely that the absence of the announcement is due to the access point not implementing announcements, or network selection via EAPOL. However receipt of a network announcement that does not acknowledge the NID requested in an EAPOL-Start can indicate that the EAPOL-Start was lost (most probably due to the access point's Common Port becoming MAC_Operational slightly after that of the accessing system as a link comes up), or that the number of virtual ports available has been exceeded.

Table 10-1—Announcement performance parameters

Parameter	Value
Announcement Time (periodic transmission interval)	5 seconds
Announcement Hold Time	15 seconds
Announcement rate limit (EAPOL-Announcements per second)	5
Announcement Delay	0.03 seconds

Transmission of announcements and specific announcement parameters can be enabled or disabled per NID per port. Each PAE shall provide the capability to disable transmission of announcements.

10.3 Receiving announcements

Information received in unprotected EAPOL-Announcements could have been modified or supplied by an attacker, and is treated as a hint to speed network access rather than a definitive statement of fact. If a supplicant receives an announcement indicating that certain mechanisms (MACsec, for example) are unavailable it will still attempt to use those selected by its Logon Process controls (12.5). The absence of MKA or MKA + MACsec from an announcement, while Access Status or Unauthenticated Access report connectivity, can be taken as a hint that the supplicant enable its Controlled Port (if permitted by the Logon Process controls, and after applying the client policies appropriate to unauthenticated connectivity) before waiting for MKA to fail or to report the access point's lack of MACsec capability, but MKA is still attempted (if a CAK is available). Similar considerations apply to the use of EAP, as an EAP attempt can also be made while unauthenticated communication is in progress. Any significant disparity between the information received in an unprotected announcement and that subsequently conveyed by MKA or a secured Controlled Port could be recorded and used as a prompt to the system administrator to change the Logon Process controls so that it effectively ignores unprotected Announcements.

NOTE—EAPOL-PDUs transmitted to the specified group address are not forwarded by certain bridges, thus reducing the chance of an attack from a distant part of the network infrastructure.

Supplicants need to be aware that access points might not transmit announcements, or exclude certain announcement parameters, as a consequence of administrative security policies. Receipt of unprotected KMD information is a hint that CAKs cached with that KMD are likely to work, but absence of a KMD announcement does not preclude an attempt to use a cached CAK for that KMD.

EAPOL-Announcements are intended to communicate current state, not to transport bulk data. Each Announcement completely replaces any previously received, with the exception that Announcements received by a secured Controlled Port shall be held for Announcement Hold Time seconds (see Table 10-1) before being replaced by any more recent unprotected Announcement, and Announcements transmitted to the unicast destination MAC address of the recipient are similarly held before being replaced by more recent unprotected generic information.

A given PAE can implement both Supplicant and Authenticator functionality (application scenarios when this can be useful are described in 7.2 and 7.4), and can both make and use received announcements. However announcement information received is held quite separately from that transmitted (see Figure 12-3) and is not retransmitted.

10.4 Managing announcements

This standard does not specify how a network access point ensures that access is provided to a network or network service represented by a particular NID. Providing such access can involve the coordination of a number of policy controls (for example, including a port in a VLAN Member Set, see IEEE Std 802.1Q), and it is beyond the scope of this standard to determine whether the NIDs announced change as a result of managing those controls, or whether NIDs are managed to change those controls. However in the interests of simplifying operational practice, this standard specifies management operations that allow a system to report on and provide control over the announcements to be sent, and those received.

Transmission and reception of announcements may be disabled per port. Announcements for particular NIDs may be enabled or disabled on individual ports, but this does not preclude the use of additional system policies to determine which announcements to transmit.

The Unauthenticated Access, Access Capabilities, and KMD parameters for each transmitted announcement do not change as a result of authentication attempts, and may be configured (for each NID) for the system as a whole. The supported Ciphersuites should reflect the capabilities of the SecY supporting the port through which the announcement has been transmitted. Access Status reflects connectivity as a

result of authentication attempts, and might be set directly by the system or configured by AAA protocols (see, for example, Annex D). The MIB specified by this standard allows the announced values of Access Status to be read but not configured, as the latter would be unlikely to prove timely.

The parameters of received announcement parameters may be read by management.

11. EAPOL PDUs

This clause specifies the EAPOL PDUs exchanged between peer PAEs: to support authentication using EAP (PACP, see Clause 8); to support the MACsec Key Agreement protocol (MKA, see Clause 9); and to announce network identities and other access point capabilities. It specifies the following:

a) Rules for the transmission, addressing, and protocol identification of EAPOL PDUs (11.1)

b) Assignment of the PAE Ethernet Type, to identify EAPOL PDUs (11.1.4)

c) Rules for the representation and encoding of protocol fields as octets in EAPOL PDUs (11.2)

d) A common structure for EAPOL PDUs (11.3)

e) General rules for EAPOL PDU validation (11.4) and protocol version handling (11.5)

f) EAPOL Packet Types for use by peer client entities (Table 11-3)

g) EAPOL Packet Body validation and parameter encoding for each Packet Type (11.6–11.11)

NOTE—EAPOL is an abbreviation of *EAP over LANs*, reflecting the use of EAPOL PDUs to transport EAP packets between PAEs in a LAN environment, and to initiate and terminate EAP authentication. EAPOL PDUs also support key agreement and key exchange protocols, even when EAP authentication is not used. Use of common addressing and protocol identification ensures that the underlying connectivity for EAP transport matches that for the protocols that can use EAP results: so authentication does not result in master keys that cannot be used by a key agreement protocol because the participating systems cannot reach each other using that protocol, likewise use of a prior authentication result for key agreement does not result in secure connectivity that cannot be supported by reauthentication.

11.1 EAPOL PDU transmission, addressing, and protocol identification

EAPOL PDUs are transmitted and received using the service provided by an LLC entity that uses, in turn, a single instance of the MAC Service provided at an MSAP. Each EAPOL PDU is transmitted as a single MAC service request, and received as a single MAC service indication, with the following parameters:

a) Destination address (11.1.1)

b) Source address (11.1.2)

c) MSDU

d) Priority (11.1.3)

The MSDU of each request and indication is the EAPOL MPDU (MAC Protocol Data Unit). This MPDU comprises an Ethertype protocol identification header (11.1.4) followed by the EAPOL PDU proper (11.3).

NOTE 1—For the purposes of this standard, the term “LLC entity” includes entities that support protocol discrimination using the Ethertype field as specified in IEEE Std 802.

NOTE 2—The complete format of an EAPOL frame ‘on the wire’ or ‘through the air’ depends not only on the EAPOL MPDU format, as specified in this clause, but also on the procedures (both media access method dependent and independent) used to support the MAC Service in a particular application scenario, as specified in Clause 7. Clause 7 includes the interface stack specifications necessary for interoperability, these do not add VLAN tags to transmitted frames prior to submitting them to media access method dependent procedures unless tagging is shown explicitly.

11.1.1 Destination MAC address

The destination address for each MAC service request used to transmit an EAPOL MPDU is an individual address associated with a peer PAE or a group address, as specified for media independent operation by Table 11-4. IEEE Std 802.11 specifies the use of addresses for IEEE 802.11 PAE and key exchange operations.

Where a group destination address is used, the choice of address depends on the potential scope of the connectivity association that includes the desired peer entities. A given system can use EAPOL in connectivity associations with potentially different scopes. MACsec can secure both access to a provider network and transmission between systems attached to that network (7.7, Clause 11 of IEEE Std 802.1AE-2018). IEEE Std 802.1Q recognizes connectivity associations between peer MAC service users with the following scopes:

a) Within a LAN or VLAN that potentially encompasses the whole of a Bridged Local Area Network. Connectivity between individual LANs in the network might be supported by one or more Provider Bridged Networks or Provider Backbone Networks, but the connectivity association typically excludes systems that compose those supporting networks.

b) Within a customer's LAN and bounded by MAC Bridges, VLAN Bridges, or end stations.

c) Within a provider's LAN forming part of a Provider Bridged Network, or within a LAN providing access for a customer to a provider, and bounded by MAC Bridges, VLAN Bridges, Provider Bridges, Provider Backbone Edge Bridges, Provider Backbone Bridges, or end stations.

d) Within an individual LAN supporting the MAC service using media dependent access methods and bounded by end stations and all systems that use media independent protocols or media dependent convergence protocols to support the MAC service, including MAC Bridges, VLAN Bridges, Provider Bridges, Provider Backbone Edge Bridges, Provider Backbone Bridges, and TPMRs.

Table 11-1 summarizes the group MAC addresses that can be used as the destination address for EAPOL PDUs, including their potential scope as constrained by reserved address filtering by bridges.

Table 11-1—EAPOL group address assignments

Address assignment	Address value	Filtered by:[a]				
		EDE-CC Edge components				
			MAC Bridge & C-VLAN components[b]			
				PEB C-VLAN component w/ single PEP[c]		
					S-VLAN components	
						TPMR components
EDE-CC PEP Address[d]	01-80-C2-00-00-1F	Y				
Bridge Group Address, Nearest Customer Bridge group address[e]	01-80-C2-00-00-00	Y	Y			
EDE-SS PEP Address[e, f]	01-80-C2-00-00-0B	Y	Y	Y		
Nearest non-TPMR Bridge group address, IEEE Std 802.1X PAE address[e, g]	01-80-C2-00-00-03	Y	Y	Y	Y	
Individual LAN Scope group address, Nearest Bridge group address[e, h]	01-80-C2-00-00-0E	Y	Y	Y	Y	Y

[a] Y indicates, Yes, this address is filtered by the component.

[b] Including a C-VLAN component that supports more than one provider network service instance (multiple PEPs) in a PEB.

[c] As specified in IEEE Std 802.1Q.

[d] This address is assigned in Table 15-1 of IEEE Std 802.1AE-2018.

[e] These addresses are assigned in Table 8-1, Table 8-2, and Table 8-3 of IEEE Std 802.1Q-2018.

[f] Address filtering by an EDE-SS Edge component is specified in IEEE Std 802.1AE-2018.

[g] Identified as the Nearest non-TPMR Bridge group address in IEEE Std 802.1Q and as the IEEE 802.1X PAE address in IEEE Std 802.1Q-2003, IEEE Std 802.1Q-2005, and this standard.

[h] It is intended that no IEEE 802.1 relay device will be defined that will forward frames that carry this destination address.

A group MAC address, the PAE group address (see Table 11-1), is assigned specifically for use by EAPOL clients designed to maximize plug-and-play interoperability, and should be the default for those clients. This address has scope 11.1c), and is also referred to and used as the ‘nearest non-TPMR bridge group address’. It is useful in application scenarios 7.1 through 7.6 whether network access or infrastructure support is provided by C-VLAN, S-VLAN, or router components, while allowing TPMR-transparent operation. In application scenario 7.7, the Bridge Group Address (filtered by C-VLAN components but not by S-VLAN components) is used. Where authentication and key agreement is required between or with a TPMR, the Individual LAN Scope group address is used.

11.1.2 Source MAC address

The source address for each MAC service request used to transmit an EAPOL MPDU shall be an individual address associated with the service access point at which the request is made. In an IEEE 802.11 access point, the BSSID associated with the wireless interface is used.

NOTE—The numerical comparison of one MAC Address with another, where required by this standard, is achieved by deriving a number from the MAC Address according to the following procedure. The consecutive octets of the MAC Address are taken to represent a binary number; the first octet that would be transmitted on a LAN medium when the MAC Address is used in the source or destination fields of a MAC frame has the most significant value, the next octet the next most significant value. Within each octet, the first bit of each octet is the least significant bit

11.1.3 Priority

The priority associated with each MAC Service request should be the default associated with the service access point. Transmitted EAPOL MPDUs are not Virtual LAN (VLAN) tagged by default, but may be priority tagged. All PAEs shall be capable of receiving both priority tagged and untagged EAPOL MPDUs.

NOTE—The structure of the tag header used for priority tagging is specified in IEEE Std 802.1Q.

11.1.4 Ethertype use and encoding

All EAPOL MPDUs shall be identified using the PAE Ethertype specified in Table 11-2.

Table 11-2—EAPOL Ethernet Type assignment

Assignment	Value
Port Access Entity Ethernet Type	88-8E

Where the LLC entity uses an MSAP that is supported by a specific media access control method (for example, IEEE Std 802.3) or a media access control independent entity (for example, IEEE Std 802.1AE) that directly supports encoding of EtherTypes, the LLC entity shall encode the PAE Ethertype as the first two octets of the MPDU.

Where the LLC entity uses an MSAP that is supported directly by a specific media access control method (for example, IEEE Std 802.11) that does not directly support Ethertype encoding, the PAE Ethertype shall be encoded in the initial octets of the MPDU according to the procedures specified in IEEE Std 802 for Subnetwork Access Protocols (SNAP).

NOTE—The SNAP discriminator comprises the octets AA-AA-03-00-00-00 prepended to the PAE Ethertype.

11.2 Representation and encoding of octets

All EAPOL PDUs consist of an integral number of octets, numbered starting from 1 and increasing in the order that they are put into a MAC frame. The bits in each octet are numbered from 1 to 8, where 1 is the low-order bit. When consecutive octets are used to encode a binary number, the lower numbered octet contains the more significant bits of the binary number.

When the encoding of (an element of) an EAPOL PDU is represented using a diagram in this clause, the following representations are used:

a) Octet 1 is shown toward the top of the page, higher numbered octets being toward the bottom.

b) Where more than one octet appears on a given line, octets are shown with the lowest numbered octet to the left, higher numbered octets being to the right.

c) Within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

11.3 Common EAPOL PDU structure

The EAPOL PDU comprises the octets following the PAE Ethertype in an EAPOL MPDU. All EAPOL PDUs comprise a Protocol Version (11.3.1), a Packet Type (11.3.2), a Packet Body Length (11.3.3), and (if the Packet Body Length is non-zero) a Packet Body (11.3.4), as shown in Figure 11-11.

	Octet number
Protocol Version (11.3.1)	**1**
Packet Type (11.3.2)	**2**
Packet Body Length (11.3.3)	**3 – 4**
Packet Body (11.3.4)	**5 – (4 + Packet Body Length)**

Figure 11-1—Common EAPOL PDU structure

11.3.1 Protocol Version

This one octet field represents an unsigned binary number identifying the EAPOL protocol version supported by the transmitter. An implementation conforming to this specification shall use the value 0x03.

11.3.2 Packet Type

This field is one octet in length. Table 11-3 lists the Packet Types specified by this standard, clause(s) that specify Packet Body encoding, decoding, and validation for each type, and the protocol entities that are the intended recipients. All other possible values of the Packet Type shall not be used: they are reserved for future extensions. To ensure that backward compatibility is maintained for future versions, validation, and protocol version handling for all types of EAPOL PDUs shall follow certain general rules (11.4, 11.5).

If the recipient of an EAPOL PDU does not have a client entity appropriate to that Packet Type, as indicated by its Recipient Entity in Table 11-3, that PDU is discarded. Additional recipient entities could be specified in future revisions of this standard through the assignment of additional Packet Types in conjunction with any Protocol Version.

Table 11-3—EAPOL Packet Types

Packet Type	Value	Recipient Entity(ies)	Encoding, decoding, validation specification
EAPOL-EAP[a]	0000 0000	PAE/PACP[b]	11.4, 11.5, 11.8
EAPOL-Start	0000 0001	PAE/PACP Authenticator PAE/Logon Process	11.4, 11.5, 11.6
EAPOL-Logoff	0000 0010	PAE/PACP Authenticator	11.4, 11.5, 11.6
EAPOL-Key	0000 0011	[c]	11.4, 11.5, 11.9
EAPOL-Encapsulated-ASF-Alert	0000 0100	ASF Helper	11.4, 11.5, 11.10
EAPOL-MKA	0000 0101	PAE/KaY	11.4, 11.5, 11.11
EAPOL-Announcement (Generic)	0000 0110	PAE/Logon Process	11.4, 11.5, 11.12
EAPOL-Announcement (Specific)	0000 0111	PAE/Logon Process	11.4, 11.5, 11.12
EAPOL-Announcement-Req	0000 1000	PAE/Logon Process	11.4, 11.5, 11.13

[a]The EAPOL-EAP Packet Type was referred to as the EAP-Packet Packet Type in previous revisions of this standard.

[b]The EAPOL-EAP Packet Type does not distinguish between an Authenticator or a Supplicant as a recipient. Where both are implemented for a given PAE, each receives its own copy of the EAPOL PDU. Further processing, or discard, by the recipient EAP Higher Layer entity is as specified for EAP and the EAP methods implemented.

[c]The recipient entity for EAPOL-Key frames is determined by the Descriptor Type. See 11.8.

11.3.3 Packet Body Length

This two octet field encodes an unsigned binary number that defines the length in octets of the Packet Body field (see 11.3.4); a value of 0 indicates that no Packet Body is present.

11.3.4 Packet Body

The parameters encoded within the Packet Body field, if present, are particular to each Packet Type.

11.4 Validation of received EAPOL PDUs

A received EAPOL PDU shall be processed as specified by Table 11-3 if and only if

a) The destination MAC address of the MAC service indication is either the group address recognized by the receiving MSAP for the application scenario (see 11.1.1) or the individual MAC address of the MSAP used as the source address of transmitted MPDUs; and

b) The received MPDU contains the PAE Ethernet Type encoded as specified in 11.1.4; and

c) The received EAPOL PDU contains at least two octets, i.e., at least a Packet Type field; and

d) The Packet Type is one of the values specified in Table 11-3, and the receiving EAPOL client(s) include a protocol entity of the appropriate type as specified in Table 11-3.

e) Table 11-4 specifies that the receiving EAPOL client receives individual destination addressed frames (if the destination address of the MPDU is an individual address) or group addressed frames (if the destination address is a group address).

f) The Packet Body Length denotes a Packet Body that is contained within the octets of the received EAPOL MPDU.

Otherwise the received EAPOL PDU shall be discarded.

Any octets following the Packet Body field in the frame conveying the EAPOL PDU shall be ignored.

NOTE—The frame can contain additional octets as a result of Ethernet frame size considerations. Not all of those octets are necessarily under the control of the transmitter if VLAN tags or other tags have been added and removed.

Table 11-4—EAPOL Packet Type Destination Addressing

Packet Type	Transmission	Reception
EAPOL-EAP	Authenticator (Virtual Port): Individual[a] Authenticator (Real Port): Group Supplicant: Group	Supplicant: Individual or Group Authenticator: Individual or Group[b]
EAPOL-Start	Supplicant: Group[c]	Authenticator: Individual or Group
EAPOL-Logoff	Supplicant: Group	Authenticator: Individual or Group
EAPOL-Key	[d]	[d]
EAPOL-Encapsulated-ASF-Alert	[e]	[e]
EAPOL-MKA	KaY: Group	KaY: Group
EAPOL-Announcement (Generic)	Logon Process (Network access point): Group	Logon Process (Accessing system): Group
EAPOL-Announcement Specific)	Logon Process (Network access point): Individual	Logon Process (Accessing system): Individual
EAPOL-Announcement-Req	Logon Process (Accessing system): Group	Logon Process (Network access point): Individual or Group

[a]A network access point that is capable of providing virtual ports can transmit using an EAPOL-EAP using a Group Address prior to activating its first virtual port, effectively making use of its Real Port for that transmission. This allows the network access point to transmit when the Common Port is first enabled, without waiting for an EAPOL-Start that could have been lost if an accessing system were to set MAC_Operational first.

[b]Authenticators receive Group addressed EAPOL-EAP packets: (a) to provide compatibility with prior revisions of this standard (b) so that the future design of EAP is not constrained to exchanges initiated by an Authenticator. Authenticators also receive Individual destination addressed EAPOL-EAP packets to minimize constraints on future PACP development.

[c]This table specifies addresses for media-independent operation, IEEE Std 802.11 specifies address use for IEEE 802.11 operation. IEEE 802.11 stations can use an Individual Address for EAPOL frames (including EAPOL-Start) where this table specifies Group.

[d]As specified for the specific Key Descriptor.

[e]As specified by the DMTF, Alert Standard Format (ASF) Specification.

11.5 EAPOL protocol version handling

To ensure that backward compatibility is maintained between versions of this protocol, a version **A** protocol implementation shall interpret a received EAPOL PDU with protocol version number **B** as follows:

a) Where **B** is greater than or equal to **A**, the EAPOL PDU shall be interpreted as if it carried the supported version number, **A**, as follows:
   1) All parameters that are defined in version **A** shall be interpreted in the manner specified for version **A** of the protocol.
   2) All parameters not defined in version **A** for the given EAPOL Packet Type shall be ignored.
   3) All octets that appear in the EAPOL PDU beyond the largest numbered octet defined for version **A** for the received EAPOL Packet Type shall be ignored.

NOTE 1—As a consequence of these rules, a version 1 implementation ignores the version number. The rules allow future specification of protocol extensions, identified as new versions. Subsequent versions can be required to check the version number in order to correctly interpret the received PDU.

b) Where **B** is less than **A**, the EAPOL PDU shall be interpreted as specified for the version number, **B**, as follows:
   1) All parameters shall be interpreted in the manner specified for version **B** of the protocol.
   2) All parameters not defined in version **B** for the given EAPOL Packet Type shall be ignored.
   3) All octets that appear in the EAPOL PDU beyond the largest numbered octet defined for version **B** for the received EAPOL Packet Type shall be ignored.

NOTE 2—This edition of this standard provides all the information necessary to comply with the provisions of this subclause (11.5), without the need to consult prior editions for information on prior protocol versions.

NOTE 3—IEEE Std 802.1Xbx-2014 added support for in-service upgrades including suspension of MKA operation and recovery of the most significant bits of the PN for MACsec Cipher Suites that use Extended Packet Numbering. The EAPOL version number was unaffected by this amendment. Each MKPDU (an EAPOL PDU with a Packet Type of EAPOL-MKA) carries its own MKA Version Identifier in the Basic Parameter Set (11.11, Table 11-6, and Figure 11-8).

11.6 EAPOL-Start

Version 2 and earlier EAPOL-Start PDUs are transmitted with no Packet Body. Consistent with the protocol versioning rules (11.5), EAPOL PDUs with this Packet Type and EAP Protocol Versions are processed as normal even if they contain a Packet Body. Both the contents of the Packet Body Length field, and the contents of any Packet Body or subsequent octets are ignored.

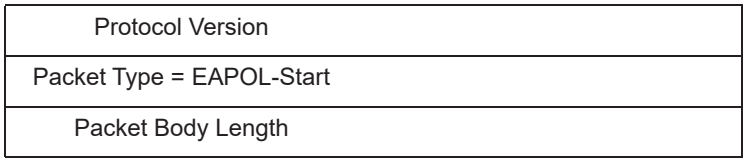


Figure 11-2—EAPOL Start-PDU (Protocol Version ≤ 2)

EAPOL-Start PDUs with Protocol Version of 3 can be transmitted with or without a Packet Body. If bit 1 (the least significant bit) of the first octet of the Packet Body of is set, receipt of the PDU solicits an announcement. The other bits in this initial octet, shall be transmitted as 0 and ignored on receipt. The remaining octets (if any) of the Packet Body encode TLVs, using the format and type codes specified for EAPOL-Announcements (11.12, Table 11-8), to convey information for the authenticator to use (and to provide to other back end services) to apply authorization and other policies.

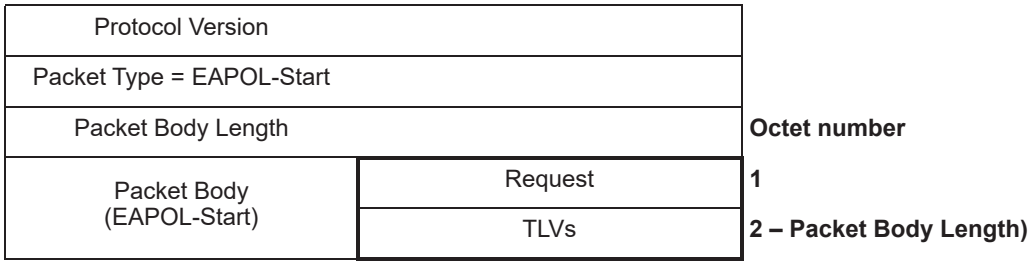


Figure 11-3—EAPOL Start-PDU (Protocol Version ≥ 3)

The first, and possibly only, TLV in an EAPOL-Start Packet Body is likely to be a NID Set TLV (11.12.1). Additional NID Set TLVs can be included in (descending) order of desirability. If the EAPOL-Start solicits an announcement, the NID Set TLV for the most desirable, recognized, NID should be included in that announcement with an Access Information TLV with the Access Requested bit set. A NID Set TLV in an EAPOL-Start-PDU can be followed by an Access Information TLV (11.12.2) specifying the Access Capabilities of the transmitting system. For some NIDs, these Access Capabilities can show that the transmitting system cannot participate in EAP. However if that is the case for the desired NID, an EAPOL-Announcement-Req should be sent instead, to avoid initiating EAP authentication with authenticators conforming to prior revisions of this standard. If an EAPOL-Start or EAPOL-Announcement-Req results in a request for a NID that lacks the necessary Access Capabilities to support authenticated connectivity, the solicited Announcement might specify an Access Status of Remedial Access for that NID.

NOTE—EAPOL start TLVs can also carry addressing information where the source MAC address of the frame is insufficient, or needs to be bound to that information, e.g., a DSL port ID.

11.7 EAPOL-Logoff

Version 3 and earlier EAPOL-Logoff PDUs are transmitted with no Packet Body. Consistent with the protocol versioning rules (11.5), EAPOL PDUs with this Packet Type are processed as normal even if they contain a Packet Body. Both the contents of the Packet Body Length field, and the contents of any Packet Body or subsequent octets are ignored.

11.8 EAPOL-EAP

The Packet Body of each EAPOL PDU with a Packet Type of EAPOL-EAP encapsulates exactly one EAP packet as shown in Figure 11-4.

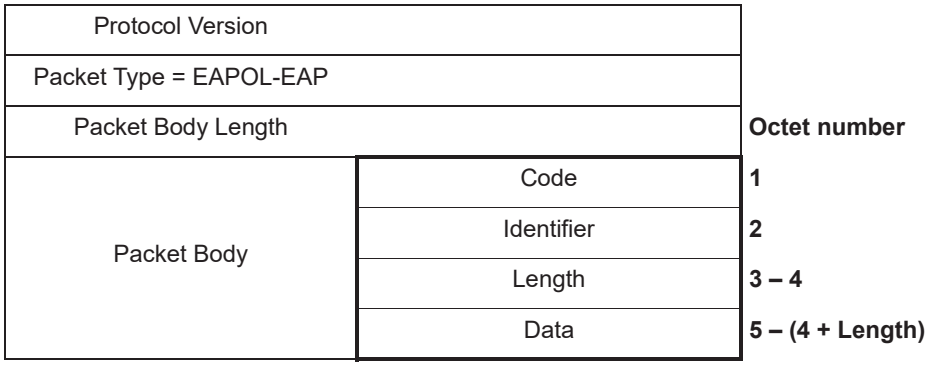


Figure 11-4—EAPOL-EAP Packet Body with EAP packet format

NOTE—This figure is purely illustrative; IETF RFC 3748 [B14] and its successor RFCs constitute the normative definition of the EAP packet formats and semantics. This standard specifies transport of EAP packets directly over the link layer medium without fragmentation, so the EAP packet size is constrained by the medium's maximum permissible frame size.

11.9 EAPOL-Key

The Packet Body of each EAPOL PDU with a Packet Type of EAPOL-Key is a Key Descriptor comprising a Descriptor Type and Descriptor Body as shown in Figure 11-5. The Descriptor Type field is one octet, representing an unsigned binary number. Table 11-5 lists the Descriptor Types specified by this standard. All other values shall not be used, and are reserved for future standardization.

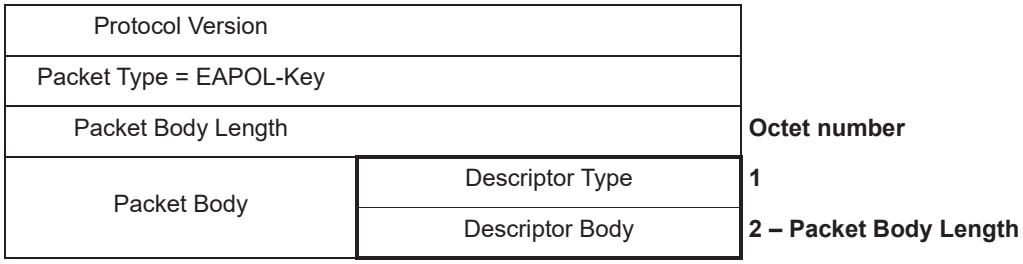


Figure 11-5—EAPOL-Key Packet Body with Key Descriptor format

The first two fields in the Descriptor Body comprise a Subtype and a Version to allow revision and extension of existing Descriptor Types without depleting Descriptor Type values. The format of these fields is not constrained by this standard, and forms part of the definition of each Key Descriptor.

Table 11-5—Descriptor Type value assignments

Assignment	Value
Deprecated[a]	1
IEEE 802.11 Key Descriptor Type[b]	2

[a]Use of this value was deprecated in IEEE Std 802.1X-2004.

[b]IEEE 802.11 Key Descriptor Type has been assigned for use by IEEE Std 802.11.

11.10 EAPOL-Encapsulated-ASF-Alert

The Packet Body of each EAPOL PDU with a Packet Type of EAPOL-Encapsulated-ASF-Alert contains exactly one ASF alert frame as specified by the DMTF Alert Standard Format (ASF) Specification.

The EAPOL-Encapsulated-ASF-Alert Packet Type supports forwarding of alerts (e.g., specific SNMP traps) irrespective of the state of authentication, authorization, or security of communication of the receiving port. All EAPOL frames received on the Uncontrolled Port with this Packet Type are passed to the protocol entity responsible for handling ASF alerts for validation and processing. This standard does not specify either the syntax or semantics of the alert messages that can be carried in this type of packet, or the protocol actions taken on receipt of a packet of this type.

11.11 EAPOL-MKA

The Packet Body of each EAPOL PDU with a Packet Type of EAPOL-MKA conveys an MKPDU. The use of MKPDU parameters by the MACsec Key Agreement protocol (MKA) is specified in Clause 9, this clause specifies their encoding. Validation checks applied to MKPDUs, beyond those specified in this clause (see Table 11-3) for EAPOL PDUs, are specified in 11.11.2.

Each MKPDU (Figure 11-6) comprises a number of parameter sets. The first of these, the Basic Parameter Set, is always present, and is followed by zero or more further parameter sets, followed by the ICV. The ICV comprises the last 16 octets of the MKPDU, as indicated by the EAPOL Packet Body Length.

NOTE 1—This standard contains a number of provisions to guard against obsolescence by future developments in cryptography, without presuming to anticipate what those developments might be. These include the ability to select different ICV algorithms and sizes. The ICV will comprise the final octets of the Packet Body, whatever its size.

MKPDU encoding, validation, and decoding follows EAPOL's versioning rules (11.2, 11.5). The Basic Parameter Set includes an MKA Version Identifier that (with other parameters in the basic set) advertises the capabilities of the transmitting MKA implementation. This information can be supplemented both by version specific parameters within the basic set and by optional sets. A consistent TLV encoding identifies each set and allows it to be skipped if unrecognized by the receiver. Addition of parameters to existing sets, and the addition of parameter sets whose support is mandatory for a given version, will be accompanied by an MKA Version Identifier increment. This standard specifies the use of MKA Version Identifier 3.

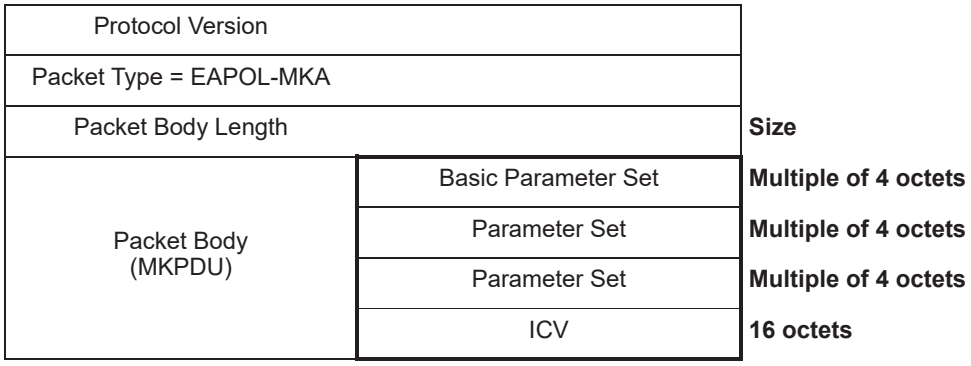


Figure 11-6—EAPOL-MKA Packet Body with MKPDU format

The parameter set encoding is designed to retain quad word octet alignment of parameters encoded in 4 or more octets, while efficiently encoding parameters that occupy only a few bits, and comprises a 4 octet header that includes a length field, a variable (possibly zero) length body, and the fewest number of null padding octets required for the entire parameter set to occupy a multiple of four octets. See Figure 11-7.

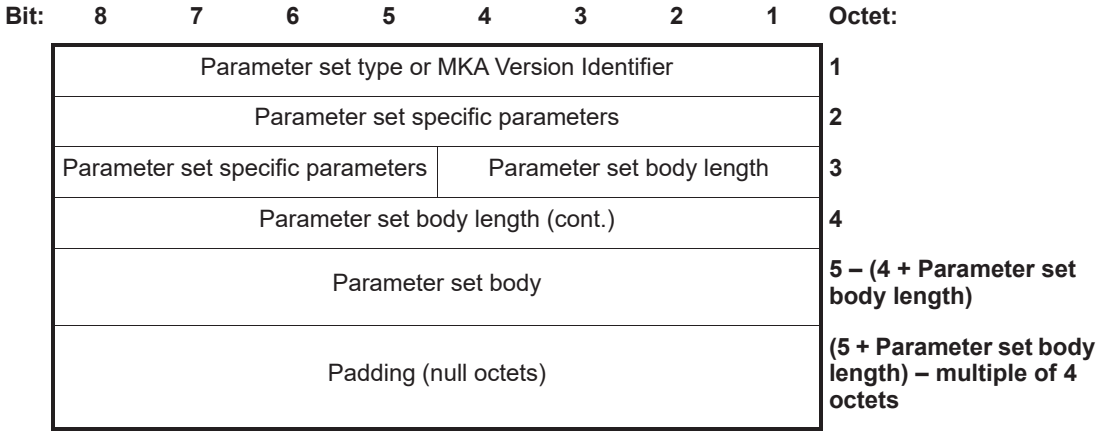


Figure 11-7—MKPDU—Parameter set encoding

NOTE 2—IEEE Std 802.1Xbx-2014 added optional support for in-service upgrades including suspension of MKA operation and recovery of the most significant bits of the PN for MACsec Cipher Suites that use Extended Packet Numbering. The MKA Version Identifier was incremented to 2 by this amendment. A single optional parameter set was added, but there were also minor changes to the behavior of the CP state machine [as a consequence of changes to the specification of the state machine interface variable chgdServer (12.2)]. Those behavioral changes removed any need for a suspended system to record the identify of the Key Server specifically, and also avoid disrupting secure connectivity if another participant that is already a CA member takes over the role of Key Server; they are transparent to other CA members that are using MKA Version 1.

NOTE 3—The MKA Version Identifier was incremented to 3 by IEEE Std 802.1Xck-2018, which did not add any new parameter sets to this standard but did impose an ordering on entries in the Live Peer List and add the Key Server SSCI to the Live Peer List parameter set (9.10, Figure 11-9).

11.11.1 MKA parameter encoding

The parameter types specified in Table 11-6 are encoded as unsigned binary numbers (11.2) within the fixed width field specified in the table. Where one or more parameters are encoded in a single octet, the bits used to encoded each parameter are specified in the figures for each parameter set, and bits in the octet that do not correspond to flags or other parameters defined for the MKPDU's MKA Version Identifier are reserved and shall be clear on transmission, i.e., shall take the value 0, and shall be ignored on receipt.

Table 11-6—MKA parameters—fixed width encoding

MKA parameter type	Field width (bits)	Notes
MKA Version Identifier	8	Encoded instead of parameter set type in Basic Parameter Set.
Flag	1	Encoded as 1 if flag is set (True).
Association Number	2	See 9.9 and IEEE Std 802.1AE.
Confidentiality Offset	2	0 if confidentiality not used, 1 if confidentiality with no offset, 2 if offset = 30, 3 if offset = 50. See IEEE Std 802.1AE.
Packet Number	32	See IEEE Std 802.1AE
MACsec Capability	2	0 if MACsec is not implemented, 1 if 'Integrity without confidentiality', 2 if 'Integrity without confidentiality' and 'Integrity and confidentiality' with a confidentiality offset of 0, 3 if 'Integrity without confidentiality' and 'Integrity and confidentiality' with a confidentiality offset of 0, 30, or 50. See IEEE Std 802.1AE.
Member Identifier	96	See 9.9
Message Number	32	See 9.4.2
Key Number	32	See 9.8
Key Server Priority	8	See 9.5
ICV	128	See 9.4.1

An SCI is encoded, as specified in Clause 9 and IEEE Std 802.1AE, in a fixed length field of eight octets. The CAK Name (CKN) (9.3.1, 6.2.2, 6.3.3) is encoded in a variable length sequence of octets within the parameter set body of the Basic Parameter Set.

Table 11-7 specifies the parameter sets defined by this revision of this standard, the format and parameters for each set are specified in Figure 11-8 through Figure 11-13. Reserved bits within octets and reserved octets are shown as 'X' in the figures.

On receipt of an MKPDU, a PAE that transmits MKPDUs with a given MKA Version Identifier

a) Shall recognize and process each parameter set specified as mandatory for that version.
b) May recognize and process parameter sets specified as optional for that version.
c) Shall ignore any parameter set that is not specified as mandatory or optional for that version.
d) Shall recognize and process each of the parameters, within each parameter set processed, that are specified as mandatory for that version.
e) May recognize and process each of the parameters, within each parameter set processed, that are specified as optional for that version.
f) Shall ignore any parameter that is not specified as mandatory or optional for that version.

NOTE—The entries in Table 11-7 follow the EAPOL protocol version handling rules (11.5).

Table 11-7—MKPDU parameter sets

Parameter set and Parameter set type		Version		Parameters	Version			Parameter specification
		1[a]	2, 3		1[a]	2	3	
Basic Parameter Set See Figure 11-8	–[b]	M	M	MKA Version Identifier	M	M	M	11.11
				Key Server Priority	M	M	M	9.5
				Key Server	M	M	M	9.5.1
				MACsec Desired	M	M	M	9.6.1
				MACsec Capability	M	M	M	9.6.1
				SCI	M	M	M	IEEE Std 802.1AE
				Actor's Member Identifier	M	M	M	
				Actor's Message Number	M	M	M	
				Algorithm Agility	M	M	M	
				CAK Name	M	M	M	9.3.1, 6.2.2, 6.3.3
Live Peer List See Figure 11-9	1	M	M	Member Identifier, Message Number tuples	M	M	M	9.4.3, 9.10
				Key Server's SSCI	—	—	M[c]	9.10
Potential Peer List See Figure 11-9	2	M	M	Member Identifier, Message Number tuples	M	M	M	9.4.3
MACsec SAK Use See Figure 11-10	3	M	M	Latest Key AN	M	M	M	9.8, 9.10
				Latest Key tx	M	M	M	9.10
				Latest Key rx	M	M	M	9.10
				Old Key AN	M	M	M	9.10
				Old Key tx	M	M	M	9.10
				Old Key rx	M	M	M	9.10
				Plain tx	M	M	M	—
				Plain rx	M	M	M	—
				Delay protect	M	M	M	9.10.1
				Latest Key Identifier (Key Server Member Identifier, Key Number)	M	M	M	9.8, 9.10.1
				Latest Key Lowest Acceptable PN	M	M	M	9.8, 9.10.1
				Old Key Identifier (Key Server Member Identifier, Key Number)	M	M	M	9.8, 9.10.1
				Old Key Lowest Acceptable PN	M	M	M	9.8, 9.10.1

Table 11-7—MKPDU parameter sets (*continued*)

Parameter set and Parameter set type		Version		Parameters	Version			Parameter specification
		1[a]	2, 3		1[a]	2	3	
Distributed SAK See Figure 11-10, Figure 11-11	4	M	M	AES Key Wrap of SAK	M	M	M	9.8
				Distributed AN	M	M	M	9.9
				Offset Confidentiality		M	M	9.7
				Key Number	M	M	M	9.8
				MACsec Cipher Suite	M	M	M	9.7
Distributed CAK See Figure 11-13	5	M	M	AES Key Wrap of CAK	M	M	M	9.5
				CA Key Name	M	M	M	9.3.1
KMD See Figure 11-14	6	M	M	KMD	M	M	M	12.6
Announcement See Figure 11-15	7	O[d]	O	Announcement TLVs	M	M	M	11.12
XPN See Figure 11-16	8	—	O[e]	MKA suspension time	—	M	M	9.18
				Latest Key: Lowest Acceptable PN (msbs)	—	M	M	
				Old Key: Lowest Acceptable PN (msbs)	—	M	M	
ICV Indicator See Figure 11-17		M[f]	M[f]	—	—			11.11.3,11.11.4

[a] M = mandatory to implement. O = optional. – = ignore on receipt.

[b] The Basic Parameter Set is identified by its position at the start of the MKPDU; the first octet encodes the MKA Version Identifier.

[c] Only encoded in MKPDUs that contain a Distributed SAK, and ignored on receipt otherwise.

[d] Mandatory to implement if EAPOL-Announcements are sent [5.10 i)].

[e] Mandatory to implement if support for Extended Packet Numbering is claimed (5.11.4).

[f] The ICV Indicator will not be encoded unless the Algorithm Agility parameter specifies the use of an ICV that is not 16 octets in length (11.11.3) and there is no requirement to implement such an algorithm; however 11.11.4 states the requirement for processing the parameter set should it be received.

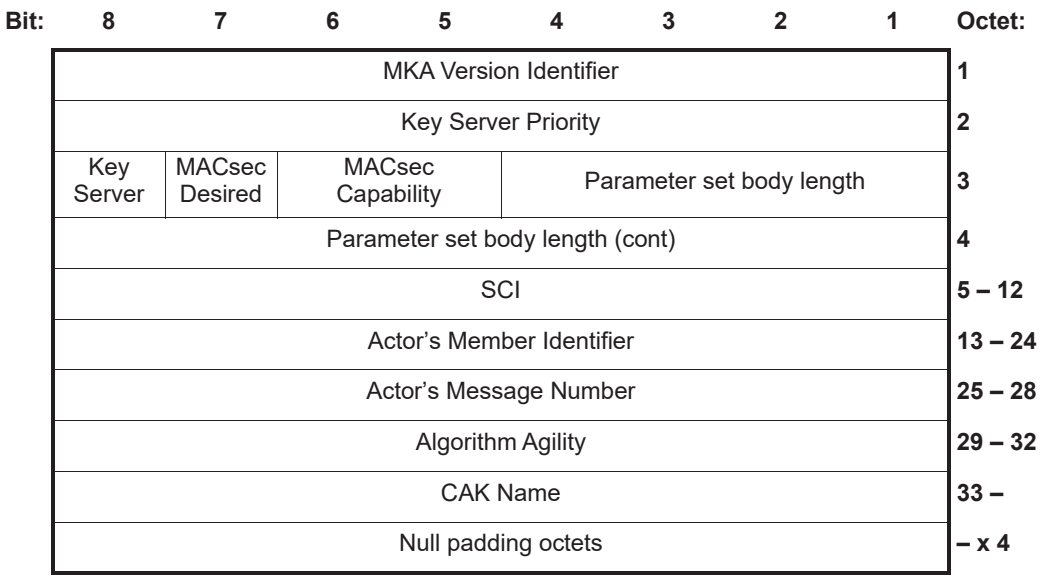


Figure 11-8—Basic Parameter Set

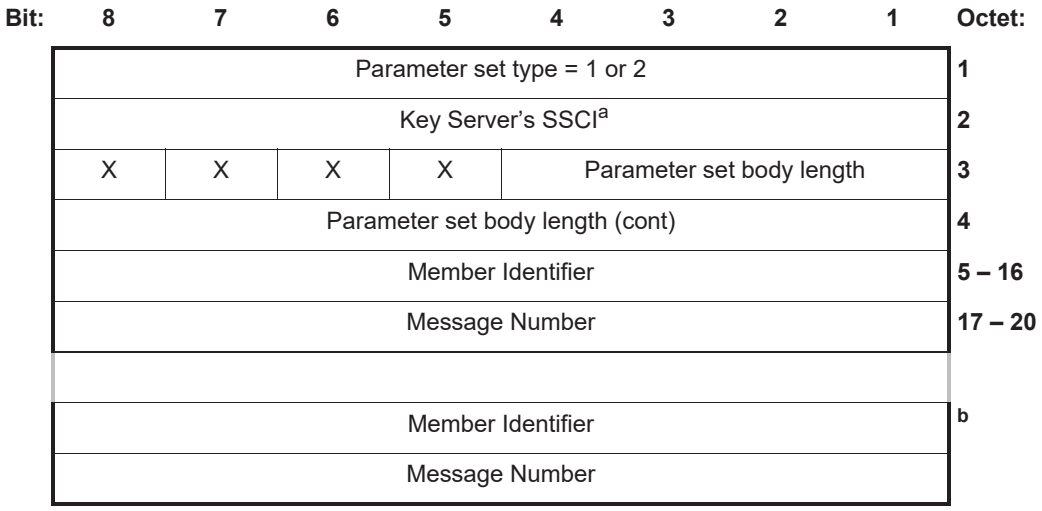


Figure 11-9—Live Peer List and Potential Peer List parameter sets

[a] The least significant octet of the Key Server's transmit SSCI is encoded in MKPDUs containing a Distributed SAK parameter set for use with an XPN Cipher Suite; otherwise, 0 is encoded.The Key Server's SSCI is distributed only in Live Peer Lists and is transmitted as zero and ignored on receipt in Potential Peer Lists.

[b] Member Identifier, Message Number tuples are repeated to the end of the parameter set.

Bit: 8	7	6	5	4	3	2	1	Octet:
Parameter set type = 3								1
Latest Key AN[a]		Latest Key tx	Latest Key rx	Old Key AN		Old Key tx	Old Key rx	2
Plain tx[b]	Plain rx[c]	X	Delay protect	Parameter set body length				3
Parameter set body length (cont)								4[d]
Latest Key: Key Server Member Identifier								5 – 16[d]
Latest Key: Key Number								17 – 20[d]
Latest Key: Lowest Acceptable PN[e]								21 – 24[d]
Old Key: Key Server Member Identifier								25 – 36[d]
Old Key: Key Number								37 – 40[d]
Old Key: Lowest Acceptable PN[e]								41 – 44[d]

Figure 11-10—MACsec SAK Use parameter set

[a]MKA uses the same AN for all the SAs for a given SAK, though IEEE Std 802.1AE does not impose this as a constraint.
[b]True if the associated Controlled Port is currently transmitting plain text, i.e., protectFrames (IEEE Std 802.1AE) is False.
[c]True if the associated Controlled Port is currently receiving plain text, i.e., validateFrames (IEEE Std 802.1AE) is not Strict.
[d]The parameter set body length will be 0 if MACsec is not supported and 40 otherwise.
[e]Least significant 32 bits if the MACsec Cipher Suite uses Extended Packet Numbering.

Bit: 8	7	6	5	4	3	2	1	Octet:
Parameter set type = 4								1
Distributed AN[a]		Confidentiality Offset		X	X	X	X	2
X	X	X	X	Parameter set body length				3
Parameter set body length (cont)								4[b]
Key Number								5 – 8
AES Key Wrap of SAK as specified in 9.8								9 – 32[c]

Figure 11-11—Distributed SAK parameter set (GCM-AES-128)

[a]Set to zero if the Key Server has decided that MACsec is not to be used. Note 0 is a valid AN.
[b]The parameter set body length will be 0 if the Key Server has decided that plain text transmission, rather than MACsec should be used, to 28 if GCM-AES-128 (the default MACsec Cipher Suite) is being used, and 36 or greater if the Cipher Suite reference number (IEEE Std 802.1AE, 14.4) is explicitly included (see Figure 11-12).
[c]Length denotes a wrapped 128-bit key.

Bit: 8	7	6	5	4	3	2	1	Octet:
Parameter set type = 4								1
Distributed AN[a]		Confidentiality Offset[b]		X	X	X	X	2
X	X	X	X	Parameter set body length				3
Parameter set body length (cont)								4[c]
Key Number								5 – 8
MACsec Cipher Suite								9 – 16[d]
AES Key Wrap of SAK as specified in 9.8								17 – 40[e]

Figure 11-12—Distributed SAK parameter set (other MACsec Cipher Suites)

[a]Set to zero if the Key Server has decided that MACsec is not to be used. Note 0 is a valid AN.

[b]Transmitted as zero and ignored on receipt if the Cipher Suite does not support Confidentiality Offset.

[c]The parameter set body length will be 0 if the Key Server has decided that plain text transmission, rather than MACsec should be used, to 28 if GCM-AES-128 (the default MACsec Cipher Suite) is being used (see Figure 11-11), and 36 or greater if the Cipher Suite reference number (IEEE Std 802.1AE, 14.4) is explicitly included.

[d]Present iff the MACsec Cipher Suite is not GCM-AES-128 (Cipher Suite reference number 00-80-C2-00-01-00-00-01).

[e]The length shown denotes a wrapped 128-bit key.

Bit: 8	7	6	5	4	3	2	1	Octet:
Parameter set type = 5								1
X	X	X	X	X	X	X	X	2
X	X	X	X	Parameter set body length				3
Parameter set body length (cont)								4
AES Key Wrap of CAK as specified in 9.8								5 – 28[a,b]
CAK Key Name								29[b] –

Figure 11-13—Distributed CAK parameter set

[a]If a future specification requires distribution of a CAK using a different key wrap or secure format an additional parameter set will be required.

[b]The length shown denotes a wrapped 128-bit key.

Bit: 8	7	6	5	4	3	2	1	Octet:
Parameter set type = 6								1
X	X	X	X	X	X	X	X	2
X	X	X	X	Parameter set body length				3
Parameter set body length (cont)								4
KMD as specified in 12.6								5 –

Figure 11-14—KMD parameter set

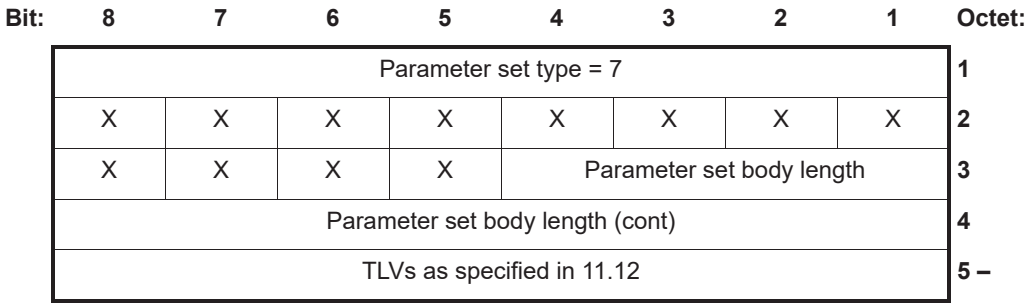


Figure 11-15—Announcement parameter set

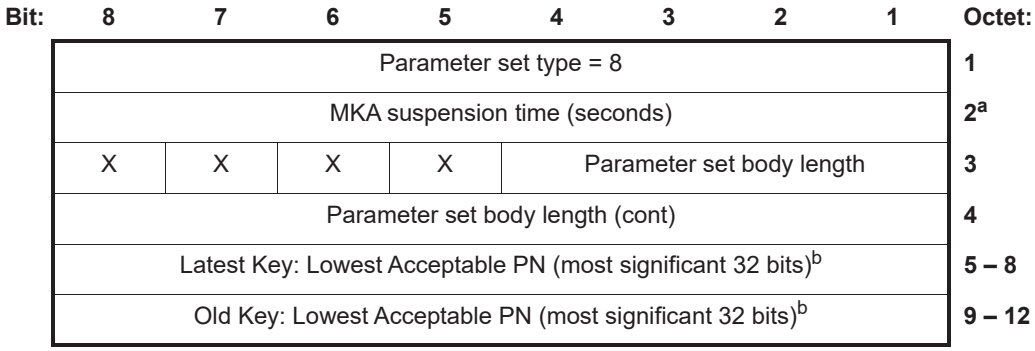


Figure 11-16—XPN parameter set

[a]The suspendedWhile timer value if the MKPDU has been transmitted by the Key Server, and suspendFor otherwise.
[b]Transmitted as zero, and ignored on receipt, if the MACsec Cipher Suite does not use Extended Packet Numbering.

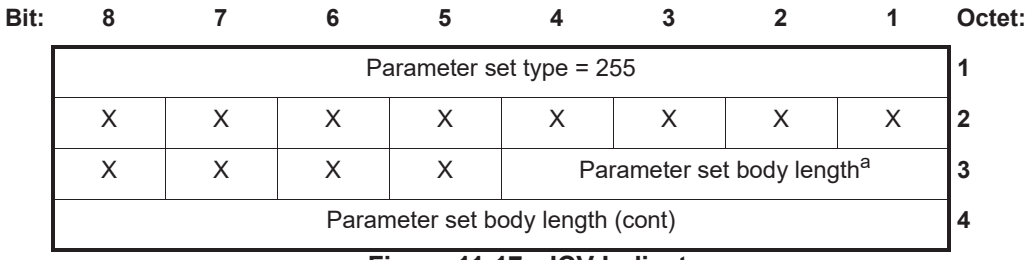


Figure 11-17—ICV Indicator

[a]The Parameter Set Body length is that of the ICV that immediately follows this parameter set and terminates the MKPDU.

11.11.2 Validation of MKPDUs

All received MKPDUs, irrespective of the conveyed value of the MKA Version Identifier, shall be discarded without ICV verification or further processing if any of the following conditions apply:

a) The destination address of the MKPDU was an individual address.

b) The MKPDU is less than 32 octets long.

c) The MKPDU is not a multiple of 4 octets long.

d) The MKPDU comprises fewer octets than indicated by the Basic Parameter Set body length, as encoded in bits 4 through 1 of octet 3 and bits 8 through 1 of octet 4, plus 16 octets of ICV.[19]

e) The CAK Name is not recognized.

Otherwise:

f) If the Algorithm Agility parameter identifies an algorithm that has been implemented by the receiver, the ICV shall be verified as specified in 9.4.1.

g) If the Algorithm Agility parameter is unrecognized or not implemented by the receiver, its value can be recorded for diagnosis but the received MKPDU shall be discarded without further processing.

Each received MKPDU that is validated as specified in this clause and verified as specified in 9.4.1 shall be decoded as specified in 11.11.4.

NOTE—Validation and verification of an MKPDU with an Algorithm Agility parameter that specifies an ICV that is not 16 octets in length is not dependent on the presence or length of the ICV Indicator (Figure 11-17).

11.11.3 Encoding MKPDUs

An implementation that transmits MKPDU PDUs with an MKA Version Identifier of 1, 2, or 3 shall encode the protocol parameters provided by the KaY as follows:

a) The Basic Parameter Set is encoded in the initial octets of each MKPDU as specified in Figure 11-8.

b) Each of the remaining parameter sets that are to be transmitted, with the exception of the Live Peer List, Potential Peer List, and ICV Indicator, are encoded as within the MKPDU in parameter set type number order (from low to high) as specified in Figure 11-10 through Figure 11-13.

c) If there are one or more Live Peers, their Member Identifier, Message Number tuples are encoded within a Live Peer List as specified in Figure 11-9. An implementation that transmits MKPDUs with an MKA Version Identifier of 3 shall order the entries in the Live Peer List and shall encode the Key Server's SSCI in Octet 2 of the Live Peer List parameter set of MKPDUs containing a Distributed SAK parameter set for use with an XPN Cipher Suite, as specified in 9.10.

d) If there are one or more Potential Peers, the Member Identifier, Message Number tuples of the Potential Peers are encoded within a Potential Peer List as specified in Figure 11-9.

e) If Algorithm Agility parameter (see Figure 11-8, 9.3.3) specifies the use of an ICV that is not 16 octets in length, the ICV Indicator is encoded as specified in Figure 11-16.

If the KaY requests a further transmission (with the same or different protocol parameters) for a given MKA Instance before an MKPDU required by this encoding has been transmitted for that instance, then the outstanding MKPDU shall not be transmitted, but the MKPDU requested in the further transmission shall be encoded for transmission as specified by this subclause (11.11).

[19]An MKPDU of less than 40 octets would be discarded, for example, if the Basic Parameter Set body length was 17.

11.11.4 Decoding MKPDUs

An implementation that transmits MKPDU PDUs with an MKA Version Identifier of 1 shall decode the protocol parameters of each MKPDU that has been successfully validated (11.4, 11.11.2, 9.4.1) as follows:

a) The Basic Parameter set is decoded from the initial octets as specified in Figure 11-8.

b) Each of the following parameter sets, if any, shall be identified by its parameter set type and decoded as specified in Table 11-7, provided that the set is completely present (as indicated by its body length parameter) within the MKPDU prior to the ICV and the parameter set body length is:

   1) A multiple of 16 octets, if the parameter set is a Live or Potential Peer List.

   2) 0, 40, or more octets, if the parameter set is the MACsec SAK Use parameter set.

   3) 0, 28, 36, or more octets, if the parameter set is the Distributed SAK parameter set.

   4) 28 or more octets, if the parameter set is the Distributed CAK parameter set.

   5) 5 or more octets, if the parameter set is the KMD parameter set.

   Otherwise the parameter set shall be discarded, and the parameters that it contains shall be ignored.

c) If the ICV Indicator is present, i.e., its initial octet is present where a parameter set type is expected, all further parameter sets in the MKPDU (if any) are ignored.

d) Additional occurrences in an MKPDU of a given parameter set (as identified by the parameter set type) and the parameters that it contains shall be ignored, and only the first occurrence processed.

11.12 EAPOL-Announcement

This subclause (11.12) specifies the encoding, validation, and decoding of EAPOL-Announcement parameters consistent with the versioning rules for EAPOL PDUs (11.5). The EAPOL Packet Type can denote either a Generic or a Specific announcement (see Table 11-3). Parameter use is specified in Clause 10.

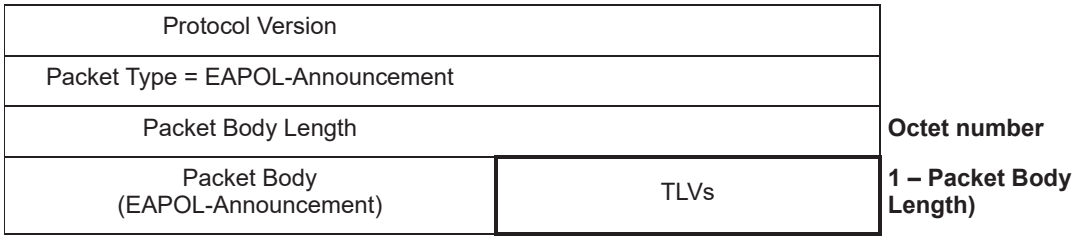


Figure 11-18—EAPOL-Announcement

Each EAPOL-Announcement (Figure 11-18) comprises a number of TLV encoded parameters. Each TLV is encoded using the format specified for IEEE Std 802.1AB, and shown in Figure 11-19. The TLV type field occupies the seven most significant bits of the first octet. The least significant bit in the first octet is the most significant bit of the TLV information string length field. The information string comprises up to 511 octets. The TLVs defined in this specification are summarized in Table 11-8.

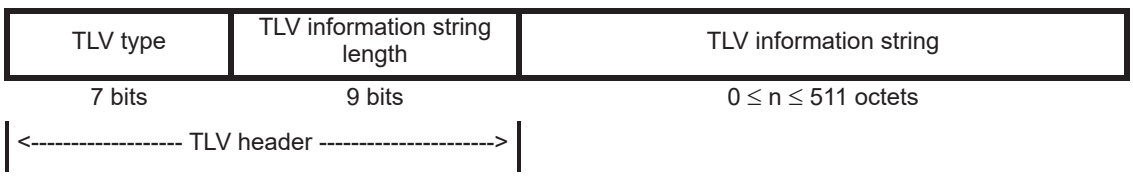


Figure 11-19—EAPOL-Announcement TLV format

Table 11-8—EAPOL-Announcement TLVs

TLV type	TLV name	Set	Validity[a]	Version 3[b]	Reference
0–110	Individual TLVs reserved for future standardization	No	Reserved for future standardization	—	
111	Access Information	No	Announcement, Announcement-Req, EAPOL-Start: Global, NID Set	M	11.12.2
112	MACsec Cipher Suites	No	Announcement: Global, NID Set	M	11.12.3
113	Key Management Domain	No	Announcement: Global, NID Set	M	11.12.5
114	NID (Network Identifier)	NID Set	Announcement, Announcement-Req, EAPOL-Start	M	11.12.1
115–125	Set TLVs reserved for future standardization	Yes	To be specified	—	
126	Organizationally Specific Set TLV	Yes	Specified by administering organization	O	11.12.5
127	Organizationally Specific TLVs	No		O	11.12.5

[a]Specifies the EAPOL Packet Types:Set(s) in which a given TLV is valid.
[b]If Announcements claimed for EAPOL Protocol Version 3: M–mandatory to implement, O–optional,— ignore

While the EAPOL-Announcement TLV format follows IEEE Std 802.1AB—allowing future versions of this standard to use previously defined information elements—the type space is distinct. Only TLVs designated for use in announcements shall be transmitted in EAPOL-Announcement PDUs. This standard extends the format to allow TLVs to be grouped: designating specific type values as set types. A set type TLV identifies a set and introduces TLVs that belong to that set: the set is terminated by the next occurrence of a set type TLV or by the end of the PDU's Packet Body, i.e., no further level of grouping of TLVs within a set is provided. TLVs encoded prior to any set type TLV are designated Global. Information included in a specific TLV within a TLV set takes precedence over information specified in a Global TLV. Organizationally specific TLVs may be defined, without further change to this standard, by using the Organizationally Specific TLV (127) or the Organizationally Specific Set TLV (126), depending on whether an individual information element or a set is required. The format of these TLVs is identical to that of Organizationally Specific TLV (127) defined in 9.6 of IEEE Std 802.1AB-2005, and similar considerations apply to their use (see 11.12.5).

Announcement TLVs can also be conveyed in MKPDUs (11.11), providing confirmation that information received in an unprotected EAPOL-Announcement was not supplied or modified by an attacker. The NID Set and other Announcement TLV information can also be conveyed with an EAPOL-Start (11.6) (if permitted by the Validity column in Table 11-8).

11.12.1 Network Identity (NID) Set TLV

The NID Set TLV (see Figure 11-20) identifies a network or network service profile and can introduce a set of TLVs that provide further information about that network.

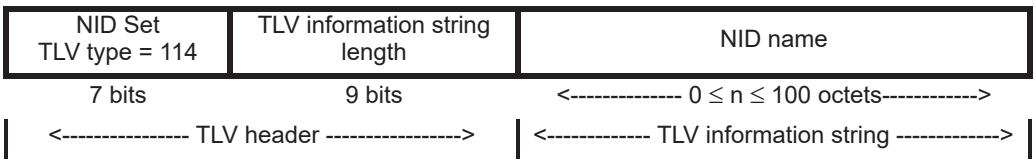


Figure 11-20—NID Set TLV format

The NID name is a UTF-8 encoded string and is intended to be globally unique, though a null name can be advertised where a network administrator wishes to announce information associated with an access point without naming or revealing a name for the network. The maximum length of a NID name is 100 octets.

11.12.2 Access Information TLV

The Access Information TLV provides the information specified in 10.1 (see Table 11-9) for a particular NID or for the port in general (if the TLV is encoded before any set TLV).

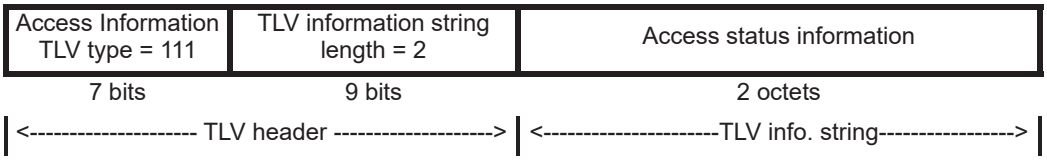


Figure 11-21—Access Information TLV format

NOTE—The Announcement can contain additional TLVs that refine or override Access Status. The use of (extended) RADIUS attributes to communicate information to the authenticator for announcement is one possibility.

Table 11-9—Access Information

Information	Field	Indicates
Access Status	Octet 1: bits 1 (l.s.b) - 2	No Access (0); Remedial Access (1); Restricted Access (2); Expected Access (3)
Access Requested	Octet 1: bit 3	Set if access requested for this NID
Unauthenticated Access	Octet 1: bits 4 - 5	No Access (0); Fallback Access (1); Limited Access (2); Open Access (3)
Virtual Port Access	Octet 1: bit 6	Set if access can be provided by a virtual port.
Group Access	Octet 1: bit 7	Set if access can be through a Group CA
Reserved	Octet 1: bit 8	Reserved for future standardization. Encode as zero.
Access Capabilities	Octet 2: bit 1 (l.s.b)	EAP
	Octet 2: bit 2	EAP + MKA
	Octet 2: bit 3	EAP + MKA + MACsec
	Octet 2: bit 4	MKA
	Octet 2: bit 5	MKA + MACsec
	Octet 2: bit 6	Higher Layer (WebAuth)
	Octet 2: bit 7	Higher Layer Fallback (Webauth)
	Octet 2: bit 8 (m.s.b)	Vendor specific

11.12.3 MACsec Cipher Suites TLV

The MACsec Cipher Suites TLV (Figure 11-22) contains a list of one or more Cipher Suites supported by the system (for access to the specified network if within a NID Set) transmitting the announcement. Each Cipher Suite in the list is represented by its 8 octet Cipher Suite reference number as specified by Clause 14 of IEEE Std 802.1AE-2018. A 2 octet Cipher Suite dependent implementation capability field precedes each Cipher Suite reference number. The two least significant bits of the implementation capability field encode the MACsec Capability parameter specified in Table 11-6 and the fourteen more significant bits are transmitted as 0 and ignored on receipt. If the Authentication Requirements TLV specifies support for MACsec, and the MACsec Cipher Suites TLV is not present for a given NID, or the TLV information string length is not a multiple of 10 octets, the recipient can assume that any Global MACsec Cipher Suites TLV applies to that NID. If no MACsec Cipher Suites TLV is encoded the recipient of the Announcement can assume that the Default Cipher Suite (specified in IEEE Std 802.1AE) is supported, with both integrity protection (without confidentiality) and integrity with confidentiality (with a confidentiality offset of 0), and is the only MACsec Cipher Suite supported. GCM-AES-XPN-128 and GCM-AES-XPN-256 do not support a confidentiality offset of other than 0.

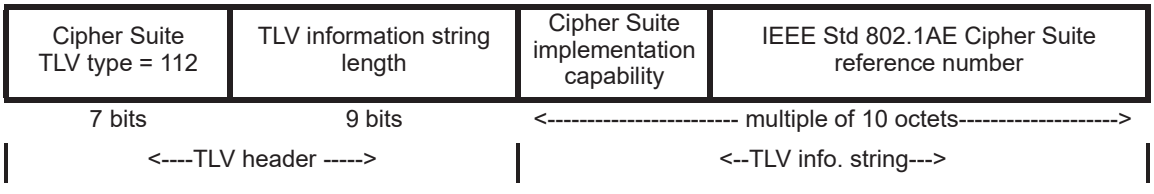
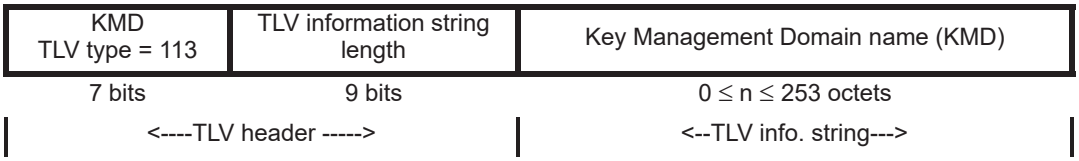


Figure 11-22—MACsec Cipher Suites TLV format

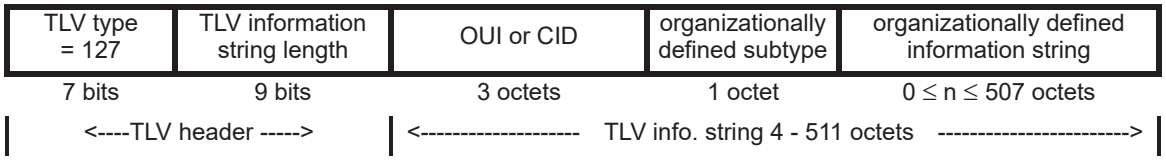
11.12.4 Key Management Domain TLV

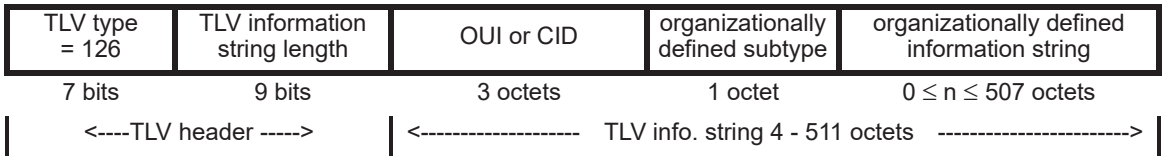
A Key Management Domain TLV (Figure 11-23) contains a string of up to 253 UTF-8 characters that names the transmitting authenticator's key management domain (6.2.3, 6.3.5). A null (zero length) KMD implies that an authenticator does not cache keys for a NID, but the absence of a KMD TLV does not.



11.12.5 Organizationally Specific and Organizationally Specific Set TLVs

Figure 11-24 and Figure 11-25 illustrate the basic format for Organizationally Specific (TLV type = 127) and Organizationally Specific Set (TLV type = 126) TLVs. The organizationally unique identifier field shall contain the defining organization's OUI or CID as assigned to that organization by the IEEE Registration Authority. The organizationally defined subtype field shall contain a value assigned by the defining organization, unique within the scope of subtypes defined for use with IEEE Std 802.1AB and this specification.





TLVs for use with IEEE Std 802.1AB need to be defined in a way that conforms to the provisions of that standard. All organizationally defined TLVs for use with EAPOL-Announcements and the MKPDU Announcement parameter set

a) Shall not solicit a response, and shall be independent of information in received TLVs.
b) Shall not be concatenated with information transmit in other TLVs to enable transmission of messages that would not fit within a single TLV.

11.12.6 Validation of EAPOL-Announcements

All received EAPOL-Announcements shall be discarded without further processing if any of the following conditions apply:

a) The destination address of the EAPOL-Announcement was neither:
   1) The individual address associated with the receiving PAE; nor
   2) The group MAC address selected for frame transmission by the receiving PAE (see 11.1).
b) The EAPOL-Announcement is less than 3 octets long.

Otherwise the received EAPOL-Announcement shall be decoded as specified in 11.12.8.

NOTE—An EAPOL-Announcement with no EAPOL Packet Body (an EAPOL Packet Body Length of zero) contains no TLVs, and will be effectively discarded since it contains no information.

11.12.7 Encoding EAPOL-Announcements

Protocol parameters for each announcement (see Clause 10) are encoded in successive octets of single EAPOL-Announcement PDU (Figure 11-18) as follows:

a) Any parameters that are Global, i.e., not specific to a particular NID Set or other Set TLV are encoded. The order of encoding is not significant. Any given TLV shall not be encoded more than once in a Global context.
b) If NID Set specific or other Set TLV specific parameters are to be encoded, the appropriate Set TLV parameter is encoded, followed by the individual TLVs that to be advertised for that Set. The order of TLV encoding within a Set is not significant. Any given TLV shall not be encoded more than once within a given Set.
c) The prior step is repeated if parameters for another Set are to be advertised.

11.12.8 Decoding EAPOL-Announcements

An implementation that transmits EAPOL-Announcements with a Protocol Version of 3 or higher shall decode the protocol parameters of each received EAPOL-Announcement that has been successfully validated (11.4, 11.12.6) in a way that produces the same results as the following procedure:

a) The TLVs are decoded in the order they are encoded in the EAPOL-Announcement.
b) If the length of any TLV would cause it to extend beyond the of the Packet Body length, decoding of the PDU shall be terminated, though already decoded TLVs (if any) shall be retained.
c) All TLVs designated as Set TLVs in Table 11-8 shall be recognized as Set TLVs (including those reserved for future standardization).
d) If an individual TLV is not recognized or will not be used, that TLV shall be skipped and the next TLV decoded, if any remain. If a Set TLV is not recognized or will not be used, TLVs are skipped until the next Set TLV or the end of the EAPOL-Announcement is encountered.
e) If the first TLV decoded is not a Set TLV, it shall be decoded and recorded as Global. Further TLVs shall also be decoded and recorded as Global (or discarded if they are only valid as TLVs of one or more particular Sets, see Table 11-8) until the first Set TLV (if present) is encountered.
f) Once a Set TLV is encountered subsequent TLVs are assigned to the identified Set until the next Set TLV is encountered.
g) If a particular TLV is encountered more than once as Global, or more than once within an identified Set, only one of its values shall be recorded (for Global, or for the identified Set as appropriate). It is undefined whether the recorded value is to be the first, last, or any other encountered by the decoder.

11.13 EAPOL-Announcement-Req

EAPOL PDUs with a packet type of EAPOL-Announcement-Req and an EAP Protocol Version of lower than 3 shall not be transmitted and shall be discarded on receipt.

EAPOL PDUs with a packet type of EAPOL-Announcement-Req and an EAP Protocol Version of 3 can be transmitted with or without a Packet Body (Figure 11-26). If no Packet Body is present or bit 1 (the least significant bit) of the first octet of the Packet Body is set, receipt of the PDU solicits an announcement. Otherwise the EAPOL-Announcement-Req can be used simply to convey network selection parameters, in the same way as an EAPOL-Start. The other bits in this initial octet, shall be transmitted as 0 and ignored on receipt. The remaining octets (if any) of the Packet Body encode TLVs, in the same way as an EAPOL-Start (11.6) but without necessarily initiating EAP.

Protocol Version	3	
Packet Type = EAPOL-Announcement-Req	0000 1000	
Packet Body Length		**Octet number**
Packet Body	Request	**1**
	TLVs	**2 – Packet Body Length)**

Figure 11-26—EAPOL-Announcement-Req (Protocol Version = 3)

NOTE—While an EAPOL-Start can be used to solicit an announcement, the EAPOL-Announcement-Req is provided for use by PAEs that do not wish to initiate EAP Authentication at the same time.

12. PAE operation

This clause describes the operation of the PAE in detail, including provisions for interoperability across a wide range of application scenarios and port-based network access control system capabilities.

The PAE's functionality, and its internal and external interfaces to functionality specified elsewhere, is modelled in terms of state machines, state variables, and procedures. This model of operation is simply a specification of the necessary functionality, and does not constrain real implementations; these can adopt any internal model of operation compatible with the externally visible behavior specified by this standard. Conformance to this standard is purely in respect of observable protocol. The notational conventions used in the state machine specification are described in Annex C.

12.1 Model of operation

Figure 12-1 provides an overview of the principal state machines and entities that model the behavior of the PAE. A PAE that is capable of instantiating virtual ports is required to implement multiple instances of one or more of the state machines depicted, and to coordinate their operation, in addition to instantiating the virtual ports themselves. The additional capabilities required to support virtual ports are detailed after the functionality required to support a single port is specified.

The Controlled Port (CP) state machine (Figure 12-2) is responsible for asserting the controlledPortEnabled signal (10.7.5 of IEEE Std 802.1AE-2018) that the PAE uses to control the MAC Operational status of the Controlled Port. When controlledPortEnabled is false, the client of the Controlled Port can neither receive nor transmit. The CP also controls the portValid signal, setting it true when communication through the port is secured by MACsec (to the extent controlled by the SecY control variables macsecProtect, macsecValidateFrames, and macsecReplayProtect see 12.2, Figure 12-2).

The CP state machine is capable of controlling a SecY (IEEE Std 802.1AE) or a PAC (6.4). The CP supports interoperability with unauthenticated systems that are not port-based network access control capable, or that lack MKA. When the access controlled port is supported by a SecY, the CP is capable of controlling the SecY so as to provide unsecured connectivity to systems that implement a PAC.

The Key Agreement Entity (KaY) manages the operation of zero or more MKA instances, each identified by a CKN and using a specified CAK, as specified in Clause 9. The KaY's participation in a given instance is represented by an actor created by the Logon Process (9.14, 12.2, 12.5). An actor is considered successful if it has one or more live partners, and the actor or one of those partners has been elected Key Server for the MKA instance. The KaY can have multiple successful actors at any one time, but only one of these can be selected as the KaY's principal actor, responsible for controlling the associated PAC or SecY. The KaY determines the candidates for selection by comparing the priority of each of the Key Servers elected by successful actors. One of those Key Servers will have the highest priority. The candidates for selection as principal actor are limited to those successful actors that have elected that highest priority Key Server as their Key Server. If the KaY is that highest priority Key Server, it can choose any one of those candidates as its principal actor, and use it to distribute SAKs. If the KaY is not that highest priority Key Server, then its principal actor will be the candidate that has most recently received a Distributed SAK parameter set (see Figure 11-11, Figure 11-12) from that Key Server. Distributed SAK parameter sets received by actors that are not candidates for principal actor are ignored. The highest priority Key Server can change its principal actor (9.5), causing its peer KaYs to change their principal actors when a Distributed SAK parameter set is received.

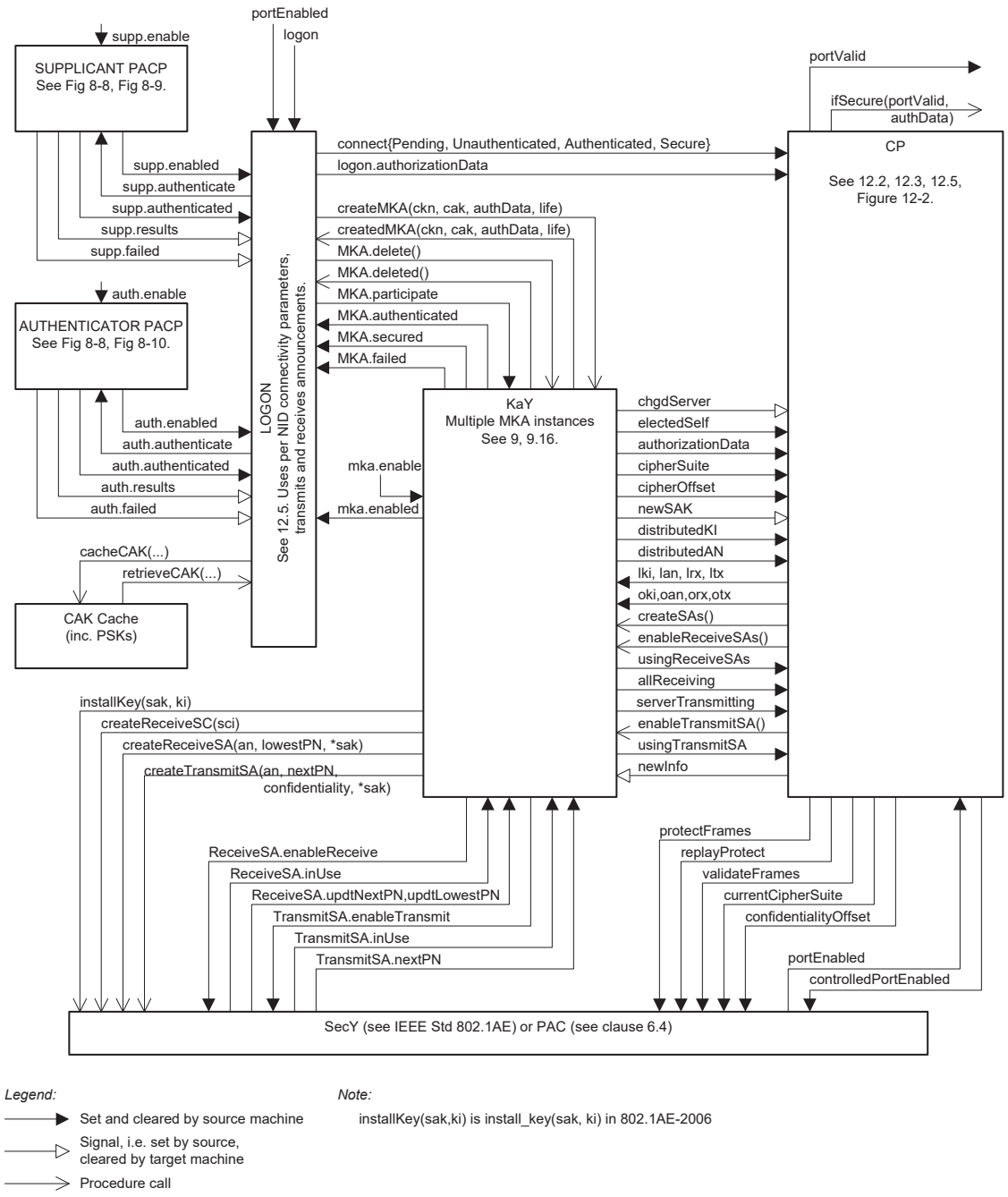


Figure 12-1—PAE state machines—overview and interfaces

MKA.secured (see 12.2) will be set iff the Key Server's principal actor has decided that MACsec is to be used and MKA.authenticated will be set iff it has decided on plain text transmission (Figure 11-12). The KaY will also report MKA.failed for any CKN for which it fails to find a partner or a Key Server (see 9.14). The Logon Process is also responsible for initiating EAP authentication as a Supplicant and or Authenticator if those capabilities are present and enabled, interfacing to the PACP state machines as specified in 8.4. CAKs that have been acquired from EAP may be cached in the CAK Cache. The CAK Cache may also be configured with one or more Pre-Shared Keys (PSKs). The Logon Process is thus responsible for the acquisition, use, and retention of all CAK, CKN tuples and for deciding (if necessary) that authentication or use of a mutual proof of prior authentication is not possible and selecting unauthenticated connectivity—

provided that is permitted by CP controls. A pre-shared CAK, CKN and or CAK, CKN tuples from previously successful authentications can be used by the KaY at the same time as a fresh authentication attempt is made using EAP, or the latter can be delayed in anticipation that prior authentication result can be used. The choice of CAK, CKN tuples by the Logon Process may be guided by Network Identity information made available through EAPOL.

In addition to reporting on the success or failure of each MKA instance, i.e., attempts to use a particular CAK, CKN tuple, to the Logon Process, the KaY interfaces to the CP, notifying the latter of changes in the Key Server, Cipher Suite, and SAK (as identified by the Key Identifier, KI). The KaY client interface (12.2) provides the CP with procedures to manage the use of SAs. These in turn use the layer management interface (LMI) provided by the SecY (see IEEE Std 802.1AE).

The real port's PAE's EAPOL Transmit and Receive Process transmits and receives all EAPOL PDUs used by the Supplicant and Authenticator PACP state machines (Clause 8), the KaY (Clause 12), the Logon Process (12.5), and to convey Announcements (Clause 10), as specified in 12.8. This process can instantiate virtual ports, using the criteria described in 12.7.

12.2 KaY interfaces

The interface provided by the KaY to the Logon Process comprises the following variables and procedures:

— mka.enabled: Set by MKA if it is operational: enabled will be FALSE if the functionality provided by the PAE is not available, or not implemented, or the control variable mka.enable (see 9.16) has been cleared by management.

— createMKA(ckn, cak, authData, life): Called by the Logon Process to create an MKA actor with the given CAK and CKN. The authData parameter specifies the authorization data, if any, associated with the actor. The life parameter specifies the maximum lifetime of CAK and CKN.

  An implementation specific means of identifying each MKA actor created is assumed by this specification. An independent instance of each of the following variables and procedures, with a name beginning 'MKA', exists for each actor.

— MKA.created(ckn, cak, authData, life): Called by MKA to notify the Logon Process that it has created a group actor, following distribution of a CAK, CKN tuple by the elected Key Server.

— MKA.delete(): Called by the Logon Process to delete (kill) the actor.

— MKA.deleted(): Called by MKA to notify the Logon Process that it has deleted the actor, either because its life has expired or because it has been replaced, as principal actor, by a new principal actor without a change of elected Key Server.

— MKA.participate: Set by the Logon Process to require the actor to be an active participant in MKA. When set MKPDUs will be transmitted even if none are received and even if MKA.failed is set. When MKA.participate is not set, the actor will transmit only for a period of MKA Lifetime following the receipt of an MKPDU from a feasible partner (9.4.6).

— MKA.authenticated: Set by MKA to indicate that the actor is the principal actor, i.e., is participating in the MKA instance that has elected the highest priority Key Server, and that Key Server has proved mutual authentication but has determined that Controlled Port communication should proceed without the use of MACsec (see 9.16).

— MKA.secured: Set by MKA to indicate that the actor is the principal actor, i.e., is participating in the MKA instance that has elected the highest priority Key Server, and that Key Server has specified the use of MACsec to secure communication (see 9.16).

— MKA.failed: Set by MKA to indicate that the actor has failed, i.e., the actor and its live partners (if any) do not include a participant willing to act as a Key Server. If the actor has failed but has not been deleted, it might receive further MKPDUs indicating that there is a potential partner and Key Server for the participants. In that case MKA will clear failed.

If in-service upgrades are supported (5.11.4, 9.18), and a suspension is in progress, the KaY will not reset MKA.secured (if set) until the suspension has been terminated (9.18.4). As a consequence (in the absence of further management changes, such as modification of the policy controls permitting connectivity) the Logon Process will not change the value of the connect signal to the CP state machine for the duration of the suspension and the MKA instance's Key Server (or its substitute) will not generate a fresh SAK, allowing the latter to remain in the CP:RETIRE or CP:SECURED states and provide continued data connectivity.

The following variables are set by the KaY's principal actor to convey information to CP:

- chgdServer: Set when a new Key Server, i.e., one whose SCI (9.4.4) was not among those used by existing SAs, has distributed an SAK. Cleared by CP.
- electedSelf: Set if the principal actor has been elected Key Server.
- authorizationData: The authorization data associated with the principal actor.
- cipherSuite: The identifier of the Cipher Suite (9.7) selected by the Key Server.
- cipherOffset: The confidentiality offset (9.7.1) selected by the elected Key Server.
- newSAK: Set when the Key Server has distributed an SAK to the principal actor.
- distributedKI: The key identifier for the last key distributed.
- distributedAN: The AN to be used with the last key distributed.
- lki, lan, lrx, ltx: The values of latest key identifier, latest association number, latest receiving, and latest transmitting fields to be included in MKPDUs transmitted by the principal actor. These fields are FALSE or zero in other transmitted MKPDUs.
- oki, oan, orx, otx: The values of oldest key identifier, oldest association number, oldest receiving, and oldest transmitting fields to be included in MKPDUs transmitted by the principal actor. These fields are FALSE or zero in other transmitted MKPDUs.

The following procedures are called by CP to convey information to the KaY's principal actor, each uses the SAK identified by the current value of distributedKI:

- createSAs(): Installs the SAK (10.7.26 of IEEE Std 802.1AE-2018). Ensures that a receive SC has been created (10.7.11 of IEEE Std 802.1AE-2018) for all live partners of the principal actor, and creates an SA with AN distributedAN for each of those SCs (10.7.13 of IEEE Std 802.1AE-2018) and for the transmit SC (10.7.21 of IEEE Std 802.1AE-2018). Note that other partners can become live subsequently, if the actor is not the Key Server, without a change of SAK. MKA will create SAs for those partners. MKA will set the value of nextPN for the transmitSA, so that PN values are not reused if the SAK has been used previously.
- enableReceiveSAs(): Enable reception for each of the SAs (10.7.15 of IEEE Std 802.1AE-2018). MKA will also enable reception for any subsequently created SAs that use the same SAK, and will also update the nextPN and lowestPN parameters for each receive SA.
- enableTransmitSA(): Enable transmission using the transmit SA (10.7.24 of IEEE Std 802.1AE-2018).

The following variables are set by the KaY's principal actor to convey information to CP, each uses the SAK identified by the current value of distributedKI:

- usingReceiveSAs: Set iff all the SAs for the SAK are capable of receiving frames, i.e., inUse is true for those SAs (10.7.14 of IEEE Std 802.1AE-2018).
- allReceiving: Set iff all the live partners of the principal actor that are capable of receiving frames [with the selected Cipher Suite and Cipher Suite capability (9.6.1)] transmitted using the transmit SA with the SAK key have indicated that reception is enabled (9.10.1).
- serverTransmitting: Set by MKA iff the elected Key Server has indicated that it is transmitting using the SAK identified by distributedKI (9.10.1).

— usingTransmitSA: Set by MKA iff the SecY is transmitting using the SAK identified by distributedKI, i.e., inUse is true for that SA (10.7.23 of IEEE Std 802.1AE-2018). Note that MKA records the final value of nextPN (10.5.2 of IEEE Std 802.1AE-2018) for the prior transmit SA, with the Key Server's member identifier (MI, 9.4.2) and the Key Identifier, to ensure that the SAK is not subsequently reused with the same PN.

The following variables are set by CP to convey information to the KaY's principal actor:

— newInfo: Set to request MKPDU transmission, and cleared by MKA following transmission. The actor can also set newinfo for its own purposes.

The CP controls the SecY using the following management controls:

— protectFrames:
— replayProtect:
— validateFrames:
— currentCipherSuite:
— confidentialityOffset:
— controlledPortEnabled:

These controls are provided by the SecY through the LMI (see IEEE Std 802.1AE). The CP's settings of protectFrames, replayProtect, and validateFrames override any settings for those parameters configured for the SecY. The KaY provides the CP, with management parameters—macsecProtect, macsecReplayProtect, and macsecValidate—that allow the initial deployment of MACsec to be managed by determining the settings of the SecY's parameters when the port is secured. The SecY's parameters can be read to determine the protection being provided at a given instant.

12.3 CP state machine interfaces

The CP state machine interfaces to the KaY as specified in Figure 12-1 and 12.2.

The following variables compose the interface provided by the CP state machine to the Logon Process:

— connect: The Logon Process sets this variable to one of the following values, to indicate to the CP state machine if, and how, connectivity is to be provided through the Controlled Port:

  Pending: Prevent connectivity by clearing the controlledPortEnabled parameter.

  Unauthenticated: Provide unsecured connectivity, setting controlledPortEnabled.

  Authenticated: Provide unsecured connectivity, setting controlledPortEnabled.

  Secure: Provide secure connectivity, using SAKs provided by the KaY (when available) and setting controlledPortEnabled when those keys are installed and in use, as specified in detail by the CP state machine.

— authorizationData: Authorization data to be made available to the client of the Controlled Port if connect is Authenticated. This standard does not specify the content or format of any authorization data provided.

Authorization data can be configured locally. In the case of successful EAP authentication as an Authenticator, the authorization data used should also reflect the set of client controls or permissions conveyed by AAA protocol acting as the EAP transport, e.g., as RADIUS attributes (see Annex D). Authorization data can be absent (null) or restricted to locally configured authorization data if AAA protocol is not used by the EAP Authenticator or if the PAE acted as a Supplicant, as existing AAA protocols do not provide authorization data to a Supplicant. When a PSK is used, locally configured authorization data (if any) is used.

The CP state machine informs the Controlled Port's client using the following variables and procedures:

— portValid: Set if Controlled Port communication is secured as specified by the MACsec control macsecProtect.

— ifSecure(portValid, authorizationData): Called prior to enabling communication through the Controlled Port. The authorizationData supplied to the client is that previously associated (when authentication occurred) with the MKA instance responsible for providing the MACsec SAKs. If MKA is not being used, but authenticated connectivity is being provided, the authorizationData is provided by the Logon Process.

NOTE—This procedure call is used to ensure that unsecured communication through the Controlled Port does not occur unless the port's client is already aware of the lack of security.

12.4 CP state machine

The PAE shall implement the function specified in the state machine contained in Figure 12-2 and the attendant definitions contained in 12.2, 12.3, and this subclause (12.4).

This standard adopts the convention of defining state machine timers as variables that are decremented, if their value is nonzero, by the operation of a state machine each time a variable tick is set. The system environment is responsible for setting this variable for each timer state machine at regular intervals. All timers used by the CP state machines have a resolution of one second; i.e., the initial values used to start the timers are integer values, and they represent the timer period as an integral number of seconds.

12.4.1 CP state machine variables and timers

The following timers are used by the CP state machine:

— transmitWhen: Ensures that a Key Server will not delay indefinitely, waiting for other participants to indicate that they are receiving, before a new SAK is used to transmit. The initial value of this timer, transmitDelay, is fixed at the value of MKA Life Time (6 s.).

— retireWhen: Imposes a delay after an new SAK is used to transmit, before reception with the prior SAK is terminated. The delay allows for reception of frames in transit from other participants, and for the fact that those participants can take longer to see that the Key Server is now transmitting with the new SAK and to enable transmission with the new SAK. The initial value of this timer, retireDelay, is fixed at 3 s.

12.5 Logon Process

The Logon Process is responsible for the managing the use of authentication credentials, for initiating use of the PAE's Supplicant and or Authenticator functionality, for deriving CAK, CKN tuples from PAE results, for maintaining PSKs, and for managing MKA instances. In the absence of successful authentication, key agreement, or support for MAC Security, the Logon Process determines whether the CP state machine should provide unauthenticated connectivity or authenticated but unsecured connectivity.

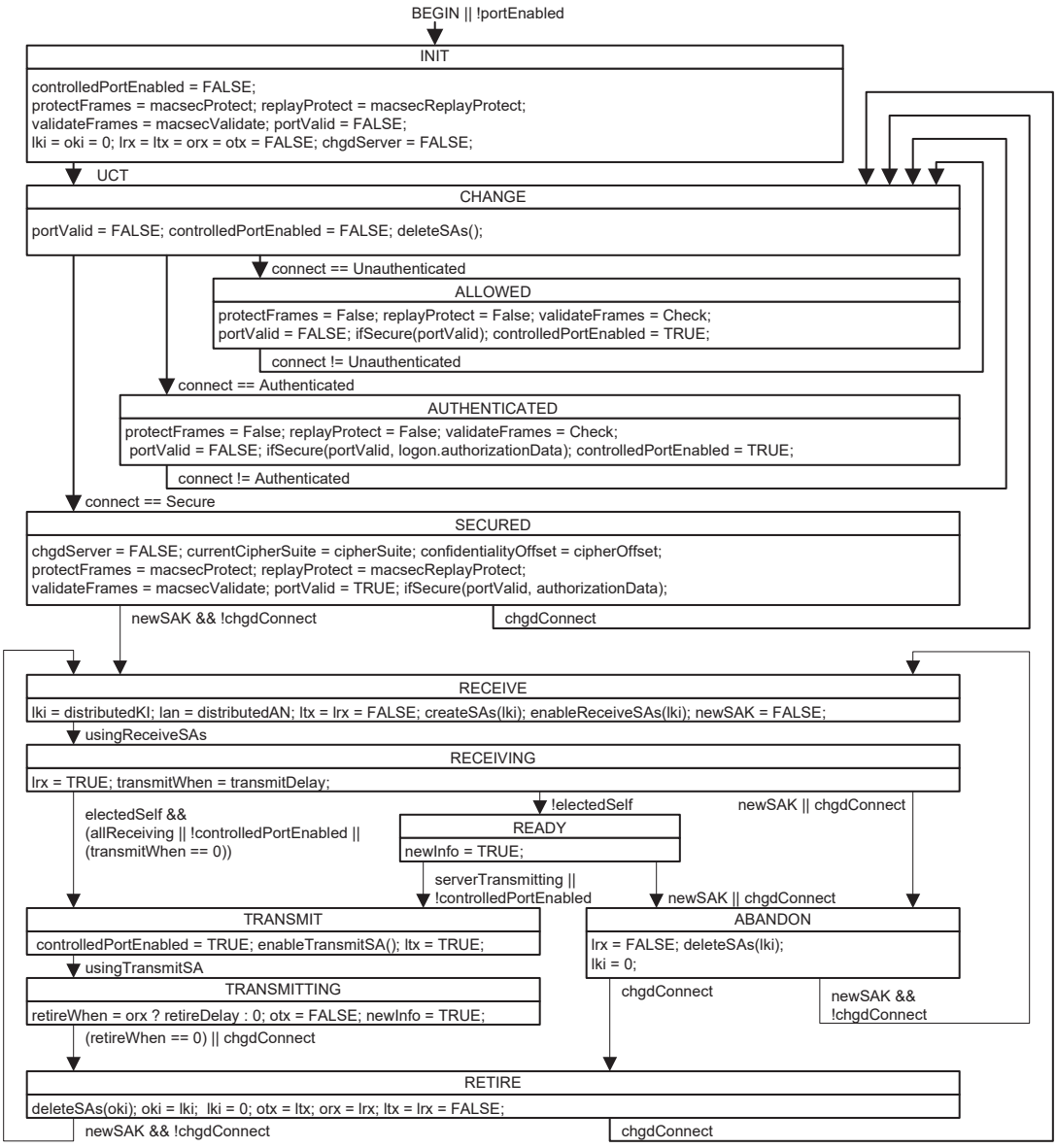
The operation of the Logon Process may be managed by the following variables:

— logon: Set if the Logon Process is to direct the PAE to provide Controlled Port connectivity. This variable is read by management, and may also be changed by management.

If logon is set, the PAE can direct the CP state machine (12.3) to provide unauthenticated, authenticated, or secure connectivity as monitored by the following variables:

— connect: See 12.3.

— portValid: See 12.1 and 12.3.



The following conditions are defined to simplify the presentation of the state machine:

chgdCipher =(currentCipherSuite != cipherSuite) || (confidentialityOffset != cipherOffset)
chgdConnect = (connect != Secure) || chgdServer || chgdCipher

Figure 12-2—CP state machine

The detail operation of the Logon Process can vary depending on the port-based network access control application, and on the capabilities supported by that implementation including, for example, network discovery and roaming (see Clause 7). This standard specifies control variables that facilitate behaviors that are potentially useful in a range of applications. Implementations may use and augment the variables specified, or may use variables specific to the implementation.

12.5.1 Controlling connectivity

When and how connectivity will be provided may be monitored, and may also be determined by changing following variables:

— useEAP: Determines when the Logon Process will initiate EAP, if the Supplicant and or Authenticator are enabled, and takes one of the following values:

Never: Never.

Immediate: Immediately, concurrently with the use of MKA with any cached CAK(s).

MKAfail: Not until MKA has failed, if a prior CAK has been cached.

NOTE—CAK, CKN tuples are cached by default and retained across transitions in portEnabled. MKA allows an authenticator to specify that a specific CAK is not to be cached.

— unauthAllowed: Determines when the Logon Process will tell the CP state machine to provide unauthenticated connectivity, and takes one of the following values:

Never: Never.

Immediate: Immediately, independently of any current or future attempts to authenticate using the PAE or MKA.

AuthFail: Not until an attempt has been made to authenticate using EAP, unless neither the supplicant nor the authenticator is enabled, and MKA has attempted to use any cached CAK (unless the KaY is not enabled).

— unsecureAllowed: Determines when the Logon Process will tell the CP state machine to provide authenticated but unsecured connectivity, takes one of the following values:

Never: Never.

Immediate: Immediately, to provide connectivity concurrently with the use of MKA with any CAK acquired through EAP.

MKAfail: Not until MKA has failed, or is not enabled.

MKAserver: Only if directed by the MKA server.

NOTE—The control provided by unsecureAllowed recognizes that there are a number of ways that authentication can occur without fully secured connectivity being immediately available. An EAP exchange could have succeeded, or MKA could have succeeded using a pre-shared or previously cached key (provided by EAP) without the CA participants selecting or being able to select use of MACsec, or both (EAP success and MKA success without MACsec).

These variables (useEAP, unauthAllowed, unsecureAllowed) may be read, and may also be changed, by management. If multiple NIDs are not supported (see below) they are associated with the null NID.

12.5.2 Active and passive participation

If an MKA instance is created (12.2) with a CAK derived from an EAP exchange and not previously cached, participate is set. If the KaY sets MKA.failed before the CAK is used successfully (setting MKA.authenticated or MKA.secured) the instance is deleted and the CAK discarded [see 9.14 i)]. If the CAK is used successfully, then it will be cached (by default) with an activate (9.16) value of Default.

If an MKA instance is created with a CAK cached with an activate value of OnOperUp, participate is set for a period of MKA Life Time and then cleared. The participant will, therefore, remain active, transmitting MKPDUs, for at least MKA Life Time, but will become passive if it does not receive an MKPDU from a feasible partner for a period of MKA Life Time.

If an MKA instance is created with a CAK cached with an activate value of Always, participate is set and is not subsequently cleared even if no MKPDUs are received. The participant will remain persistently active, transmitting MKPDUs even in the prolonged absence of a feasible partner.

If an MKA instance was created from a cached CAK that has its activate value changed from OnOperUp to Always, then participate is set. If the value is changed from Always to OnOperUp, then participate is cleared.

12.5.3 Network Identities

The Logon Process may use Network Identities (NIDs, 6.3.5, 10.1) to manage its use of authentication credentials, cached CAKs, and announcements. Its use of NIDs may be managed by the following variables:

— connectedNID: The NID (10.1, 11.12.1, 11.6) associated with the current connectivity (possibly unauthenticated) provided by the operation of the CP state machine.

— selectedNID: The NID currently configured for use by an access controlled port when transmitting EAPOL-Start frames. Defaults to the null NID.

— requestedNID: The NID (10.1, 11.12.1, 11.6) marked as Access requested in announcements, as determined from EAPOL-Start frames. Defaults to the selectedNID.

The connectedNID is determined by the PAE, and may be read by management. It can differ from both the selectedNID and the requestedNID if authenticated connectivity (either secure or unsecured) has already been established, and EAP authentication and MKA operation for both of the latter have not met the necessary conditions (as specified by the control variables unauthAllowed and unsecureAllowed).

The selectedNID may be read by management, and may be either explicitly configured by management or determined by the PAE using NID selection algorithms, not specified in this standard, that can use the recent history of authentication attempts and received announcements. If no authentication is in progress, and logon is cleared (terminating current connectivity) and then set again, connectedNID will take on the value of requestedNID (though a PAE NID selection algorithm, if used, can subsequently select another NID).

The requestedNID provides context for the PAE's EAP Authenticator, and may be read by management. If no EAPOL-Start frame has been received since the PAE's Common Port became MAC_Operational, or the last EAPOL-Start frame received for the port did not contain a requested NID, the requestedNID will take on the value of the selectedNID.

Each NID can independently specify values of useEAP, unauthAllowed, and unsecureAllowed. The unauthenticatedAccess and accessCapabilities (10.1) parameters transmitted in announcements for the NID are also independently specified, to allow their management independently of other Logon Process controls. These announcement parameters should reflect the Logon Process and other PAE management controls. Open or Limited Access should not be advertised unless unauthAllowed is Immediate. EAP and MKA should not be advertised unless the EAP Authenticator (auth.enabled, 8.4) or the KaY (9.16) is enabled respectively. However announcements can also reflect the network administrator's intent to reconfigure parameters within the authenticating system, e.g., by using RADIUS (see Annex D) to update a PVID conditionally on the outcome of EAP authentication, or to use additional system capabilities to apply other policies associated with access to the network or network service.

NIDs may be created and deleted by management. The announcement of NIDs, and the availability of the network or network service may be independently managed for each NID.

12.5.4 Session statistics

A PAE may maintain session statistics for its associated Controlled Port, suitable for communication to a RADIUS or other AAA server at the end of a session for accounting purposes (see IETF RFC 2869 [B8]), enabling access to these statistics by local or remote management before the session ends. A session begins when MAC_Operational for Controlled Port becomes true and ends when it becomes false. The counts of frames and octets can be derived from those maintained to support IETF RFC 2863 Interface MIB counters for the SecY's or the PAC's Controlled Port, but differs in that the counts are zeroed when the session begins.

Session statistics comprise the following:

— sessionOctetsRx: Session Octets Received.

— sessionOctetsTx: Session Octets Transmitted.

— sessionFramesRx: Session Frames Received.

— sessionFramesTx: Session Frames Transmitted.

— sessionId: Session Identifier. A UTF-8 string, from 3 through 253 characters long, uniquely identifying the session within the context of the PAE's system.

— sessionTime: Session Time. The duration of the session in seconds.

— sessionTerminateCause: Session Terminate Cause. The reason for the session termination, one of the following:

1) Common Port MAC_Operational FALSE.
2) systemAccess Control set to Disabled or initializePort invoked.
3) EAPOL Logoff received.
4) EAP Reauthentication Failure.
5) MKA failure or other MKA termination.
6) New session beginning.
7) Not Terminated Yet.

— sessionUserName:Session User Name. A UTF-8 string, from 0 to 253 octets long, representing the identity of the peer Supplicant.

NOTE—The Session User Name is only maintained by an EAP Authenticator, and is null (zero length) if not available.

Session statistics for a real port can be retained by the system until a new session begins on that port. Statistics for a virtual port can be retained until that virtual port is deleted or a new session begins.

12.6 CAK cache

A PAE can cache CAKs (see 6.1, 6.2.3, 6.3.5, 7.1.2, 11.12.4, 12.5, 10.1). All CAKs added to the cache have a bounded lifetime set by local policy, and are deleted from the cache when that lifetime has expired unless explicitly configured by management. PSKs can be configured in the CAK cache, or a previously cached CAK denoted as a PSK. The cache does not bound the lifetime of a PSK.

NOTE—This standard does not require the implementation of a real time clock, so a CAK's lifetime can be imprecise.

The following information shall also be associated with each cached CAK:

a) CKN (6.2).
b) A Key Management Domain (KMD). A string of up to 253 UTF-8 characters that names the transmitting authenticator's key management domain.
c) A NID. A string of up to 100 UTF-8 characters that identifies a network service (10.1). The NID can be null, and associated with a default service, if not specified in a received MKPDU.

Pairwise CAKs acquired through EAP exchanges may be cached to support rapid reauthentication or roaming. The CAK shall not be cached unless MKA has discovered a live peer with that CAK. If the CAK is cached by the EAP Authenticator, that Authenticator's MKA participant shall distribute a KMD (Figure 11-14) in the next three transmitted MKPDUs. The CAK shall not be cached by the EAP Supplicant unless it receives an MKPDU for that CAK that contains a KMD. The Supplicant can assume that the CAK will replace all prior cached CAKs for that KMD that have a matching NID (if the NID is also distributed in the MKPDU or is otherwise known) or that have a null NID (if the NID is unknown).

A Group CAK should be cached if the Key Server distributes a KMD in MKPDUs protected by that CAK. A Key Server shall not distribute a KMD for a Group CAK unless that Key Server generated the CAK.

12.7 Virtual port creation and deletion

A real port's PAE may be configured to create virtual ports to support multi-access LANs (6.3.6, 7.5, 7.6), provided that MKA and MACsec operation is enabled for that port. A PAE thus configured can maintain separate CAs for each CAK for which it is the highest priority Key Server or for which the highest priority Key Server is a distinct participant. Such CAs typically comprise only two participants, and the source MAC address of each of the participants is associated with the virtual port. An implementation that supports virtual ports shall state the maximum number of virtual ports supported for each port; if the number of possible simultaneous live CAs is greater, those with the higher priority Key Servers are maintained.

A virtual port can be created, from the point of view of PAE management (12.9), once there are live MKA participants for a CAK that meet the above criteria. A real port's PAE, that is configured as an EAP Authenticator and can create virtual ports, also will create a separate virtual port on receipt of an EAPOL-Start or EAPOL-EAP frame with a source MAC address not already associated with a virtual port. However the resources used to maintain that port should be limited to those necessary to maintain the PACP and EAP context until authentication has succeeded and an MSK is available, and to those necessary to participate in MKA with a CAK derived from the MSK until there are live participants. The creation of virtual ports may be monitored by the following variables:

— vpStart: Set if the virtual port was created by receipt of an EAPOL-Start frame.
— vpPeerAddress: The source MAC Address of the EAPOL-Start (if vpStart is set).

Prior to and while communication is secured by MACsec, the members of a CA are distinguished by their mutual possession of a CAK and the CKN that identifies that CAK. Since the CKN is conveyed in all MKPDUs, and possession of the CAK by the transmitter is verified at each receiver, the operation of MKA does not depend on the use of the destination MAC address (possibly a group MAC address) or the source MAC address to identify the virtual ports that are members or potential members of the CA. Each MKA participant encodes the SCI of the port or virtual port that it represents in each MKPDU.

Once communication has been secured by MACsec, frames transmitted by any port or virtual port can be distinguished from frames transmitted by any other port by the explicit inclusion of the SCI within the SecTAG. The format of the SCI includes a Port Identifier that allows for up to 65 535 ports to be associated with each MAC address that takes the role of the System Identifier in the SCI (see 9.9 of IEEE Std 802.1AE-2018). The MAC Address used is normally that permanently associated with the Common Port and the Port Identifier 00-01 identifies the Controlled Port and KaY permanently associated with that Common Port. Up to 65,534 virtual ports can therefore be identified for a given Common Port.

NOTE 1—Following conventions in other IEEE 802.1 standards, 00-00 is not used as a normal port identifier, and in the SCI context (IEEE Std 802.1AE) can refer to the Single Copy Broadcast (SCB) capability provided by an EPON.

Each receiving SecY knows, from the operation of MKA, what SCs it can use to receive frames, and what SAK is associated with each of the SAs for each SC. Thus inspection of the received SCI tells the SecY whether it expects to be able to deliver that frame through the virtual port's Controlled Port. Segregation of frames by virtual port depends on presence of the SecTAG and on discarding frames that are for an inappropriate SCI. The protectFrames and validate Frames controls for a SecY that supports a virtual port are always True and Strict respectively.

However the operation of higher layer protocols is intentionally independent of MACsec, except for the possible choice of policies that depend upon or mitigate the lack of secure communication. A higher layer protocol entity that relies upon MAC addresses to distinguish between its peers is not able to use more than one virtual port to communicate from one station, with a given MAC address, to another.

NOTE 2—Virtual ports use MACsec SCIs to partition connectivity into separate CAs, so connectivity can change as security is deployed, enabled, or disabled. Because this can lead to difficulties in the management of bridged networks, virtual ports and multi-access LANs (7.5) should not be used to support LANs with two or more attached bridges. They are appropriate for the attachment of end stations or hosts at the periphery of the network.

NOTE 3—If the SCI is not explicitly included in the SecTAG, frames originally transmitted by different systems directly attached to the secured LAN can be distinguished by setting the ES bit in the SecTAG and deriving the SCI by concatenating the source MAC address of the frame with the Port Identifier value of 00-01. Frames with such SecTAGs cannot be transmitted by virtual ports, and are not transmitted (though can be received) by a Network Access Point supporting a multi-access LAN. If the SCI is not explicit and the ES bit is not set, for frames transmitted on a given LAN, that LAN supports only a single point-to-point CA and virtual ports are not used with that LAN.

NOTE 4—EAPOL frames do not include additional fields to identify which of a number of possible instances of PACP and EAP should be the recipient of a given frame. Both the source and the destination of each EAPOL frame need to be identified so that an association can be formed between an instance of EAPOL in the Supplicant and an instance in the Authenticator. Those instances can then mutually authenticate and thus acquire a CAK, CKN tuple. Each of the communicating systems is identified by the MAC address associated with the Common Port, and their association can be identified by the combination of the addresses. A system that uses a number of virtual ports, each supporting PACP and EAP, uses the destination MAC address of each received frame to decide to receive that frame and the source MAC addresses of each EAPOL frame to select the recipient of that frame from amongst the instances of PACP that are supported by the same Common Port. In general this addressing does not allow multiple virtual ports on each of two stations to participate in more than one association between those systems, and thus limits the number of virtual ports associated with a station to one for each of the other stations on the same LAN. In the particular case of EAPOL the destination MAC address can be a group address so virtual ports are distinguished solely by source address, and only an Authenticator PAE can support virtual ports. In principle a Supplicant PAE could have multiple virtual ports if all Authenticator PAEs were limited to a single port per MAC Address, but this arrangement does not match application requirements.

NOTE 5—Broadcast and multicast data sent in group addressed frames transmitted by each virtual port's Uncontrolled Port can be segregated by CA by using the E bit set and C bit clear SecTAG TCI encoding (9.5 of IEEE Std 802.1AE-2018). However, this arrangement presupposes that the CA has been created with MKA operating and identifying the SCIs to be used for reception, and is therefore not available to EAPOL since the purpose of EAP is to obtain a CAK that MKA can use.

12.8 EAPOL Transmit and Receive Process

There is a single EAPOL Transmit and Receive Process for each real port. This process is responsible for multiplexing EAPOL frames transmitted by each of the components that comprise a PAE for transmission on the real port's Uncontrolled Port, and for applying the validation checks specified in 11.4 to received EAPOL frames, demultiplexing those frames to the PAE components using the EAPOL Packet Type as specified in 11.3.2 and Table 11-3.

If a real port's PAE supports virtual ports, the EAPOL Transmit and Receive Process dynamically instantiates virtual ports as described in 12.7, demultiplexing validated EAPOL frames to the appropriate PAE using the source address of the received frame. Virtual port PAEs do not include their own Transmit and Receive Process, nor do they include EAP/PACP Supplicant functionality.

12.8.1 EAPOL frame reception statistics

Any given received EAPOL frame that meets the validation criteria for destination MAC address and Ethernet Type [11.4 a) and b)] increments exactly one of the counts specified in this subclause (12.8.1). Received frames that fail to meet 11.4 a) and b) are discarded without further processing or counting.

The following counts are maintained for the Transmit and Receive Process as a whole:

— invalidEapolFramesRx: Frames that fail to meet 11.4 d).

— eapLengthErrorFramesRx: Frames that meet 11.4 d) but fail to meet 11.4 f).

NOTE 1—This count is named 'eapLength...' to align with IEEE Std 802.1X-2004, though the specification of the count in both IEEE Std 802.1X-2004 and the present revision of the standard applies to all EAPOL Packet Body length errors, not just those for the EAPOL-EAP Packet Type.

— eapolAnnouncementsRx: EAPOL Announcement frames received.

— eapolAnnouncementReqsRx: EAPOL Announcement Request frames received.

The following is maintained for the Transmit and Receive Process as a whole, if virtual ports are supported:

— eapolPortUnavailable: Frames that are discarded because their processing would require the creation of a virtual port, for which there are inadequate or constrained resources, or an existing virtual port and no such port currently exists.

The following are maintained for the Transmit and Receive Process as a whole, and for each currently instantiated PAE:

— eapolStartFramesRx: EAPOL Start Frames received.

— eapolEapFramesRx: EAPOL EAP frames received.

NOTE 2—The definition of InvalidEapolFramesRx arguably includes frames discarded if neither an Authenticator nor a Supplicant are implemented. However, it is recommended that this counter be incremented instead as IEEE Std 802.1X-2004 did not anticipate that a PAE might not implement EAP. InvalidEapolFramesRx does include EAPOL-MKA frames discarded because MKA is not implemented.

— eapolLogoffFramesRx

— eapolMKnoCKN: MKPDUs received with MKA not enabled or CKN not recognized.

— eapolMKinvalidRx: MKPDUs failing message authentication (9.4.1).

The counts reported for the Transmit and Receive Process as a whole include those for current and previously instantiated PAEs. The time at which each PAE was instantiated are recorded to assist correlation with network events.

NOTE 3—The counts can be correctly reported, without the need for each frame to increment multiple real-time counters. A count for each PAE is summed with that for the Transmit and Receive Process as a whole to respond to a request for the latter. When a virtual port is deleted its counts are added to those for the process as a whole.

12.8.2 EAPOL frame reception diagnostics

The following information is maintained for the Transmit and Receive Process as a whole:

— lastEapolFrameSource: The source MAC address of the last EAPOL frame received.

The following information is maintained for each currently instantiated PAE:

— lastEapolFrameVersion: The protocol version number of the last EAPOL frame received.

12.8.3 EAPOL frame transmission statistics

The transmission of any given EAPOL frame causes exactly one of the counts specified in this subclause (12.8.3) to be incremented.

The following counts are maintained for the Transmit and Receive Process as a whole:

— eapolSuppEapFramesTx: EAPOL-EAP frames transmitted by the Supplicant.

— eapolLogoffFramesTx

NOTE 1—Supplicants are not implemented for virtual ports or for a real port whose PAE is capable of instantiating virtual ports (see 12.7).

— eapolAnnouncementsTx

— eapolAnnouncementReqsTx

The following counts are maintained for the Transmit and Receive Process as a whole, and for each currently instantiated virtual port PAE:

— eapolStartFramesTx

— eapolAuthEapFramesTx: EAPOL EAP frames transmitted by an Authenticator.

— eapolMKAFramesTx

The counts reported for the Transmit and Receive Process as a whole include those for current and previously instantiated PAEs.

12.9 PAE management

The PAE management process controls and monitors the operation of each PAE's component protocol entities and processes, providing access to operational controls, statistics and diagnostic capabilities. Where a system is capable of supporting virtual ports, mutual authentication and or key agreement for each virtual port is modeled as being provided by a separate PAE, though a PAE that is not associated with a real port lacks the capability to dynamically instantiate virtual ports.

The management information that provides control over and reporting on each of these major capabilities (where implemented) is detailed in the relevant clause of this standard, and summarized in Figure 12-3. The containment relationships have been chosen primarily to align with object groups specified for the MIB.

NOTE—Figure 12-3 uses UML 2.0 conventions (see Fowler [B1]) together with C++ language constructs. This standard does not make extensive use of UML, but has adopted UML conventions rather than specifying its own to meet the simple needs of Figure 12-3.

Conformance to the management provisions of this standard is strictly in terms of required externally observable behavior, as revealed through the relationship of the operations supported by the SMIv2 MIB module (Clause 13) to the operation of the PAE operation to and to the specifications of protocols operated by the PAE and by other components of the system such as a SecY or PAC. There is no conformance in respect of syntactic elements used in this clause to describe interactions that are wholly contained within the system, whether those interaction occur within a given PAE or through the LMI to a component responsible for interactions made visible through a MIB via SNMP, or via another protocol.

12.9.1 System level PAE management

The following control(s) allow port-based network access control functionality to be enabled or disabled for the system as a whole:

— systemAccessControl {Enabled, Disabled}: Setting this control to Disabled deletes any virtual ports previously instantiated, and terminates authentication exchanges and MKA operation. Each real port PAE behaves as if enabledVirtualPorts was clear, the PAE's Supplicant, Authenticator, and KaY as if their enabled controls were clear, and Logon Process(es) as if unauthAllowed was Immediate. Announcements can be transmitted (subject to other controls), both periodically and in response to announcement requests (conveyed by EAPOL-Starts or EAPOL-Announcement-Reqs) but are sent with a single NID Set, with a null NID, and the Access Information TLV (and no other) with an accessStatus of No Access, accessRequested false, OpenAccess, and no accessCapabilities. The control variable settings for each real port PAE are unaffected, and will be used once systemAccessControl is set to Enabled.

— systemAnnouncements {Enabled, Disabled}: Setting this control to Disabled causes each PAE to behave as if enabled were clear for the PAE's Announcement functionality. The independent controls for each PAE apply if systemAnnouncements is Enabled.

Management information for the system as a whole is reported using the following variable(s):

— EAPOL protocol version number (11.3).

— MKA Version Identifier (11.11). Reporting version 0 indicates that MKA has not been implemented.

12.9.2 Identifying PAEs and their capabilities

Each PAE and its major components and capabilities are identified by the following variables:

— portNumber: Each PAE is uniquely identified by a port number (see IEEE Std 802.1AC). The port number used is unique among all port numbers for the system, and directly or indirectly identifies the Uncontrolled Port that supports the PAE. If the PAE has been dynamically instantiated to support an existing or potential virtual port, this portNumber, the uncontrolledPortNumber and the controlledPortNumber are allocated by the real port's PAE, and this portNumber is the uncontrolledPortNumber. If the PAE supports a real port, this portNumber is the commonPortNumber for the associated PAC or SecY.

— portType: Either RealPort or VirtualPort.

— controlledPortNumber: The port number for the associated PAC or SecY's Controlled Port.

— uncontrolledPortNumber: The port number for the associated PAC or SecY's Uncontrolled Port.

— commonPortNumber: The port number for the associated PAC or SecY's Common Port. All the virtual ports created for a given real port share the same Common Port and commonPortNumber.

NOTE—These port numbers, with the value constraints specified, not only allow each PAE and the associated ports to be identified uniquely, but also allow a network manager to locate the real port PAE responsible for instantiating a given virtual port, and the virtual ports instantiated for a given real port.

— eapolGroupAddress: The group address used for EAPOL frames (including MKA frames). For a virtual port instantiated by a real port PAE, the same as that for the real port PAE.

— implemented.supp: Set iff a PACP EAP Supplicant is implemented (Clause 8).

— implemented.auth: Set iff a PACP EAP Authenticator is implemented (Clause 8).

— implemented.mka: Set iff MKA is implemented.

— implemented.macsec: Set iff the Controlled Port is supported by MACsec.

— implemented.isupgrades: Set iff the MKA supports in-service upgrades (9.18).

— implemented.announcements: Set iff EAPOL announcement can be sent.

— implemented.listener: Set iff received EAPOL announcements can be used.

— implemented.virtualPorts: Set for a RealPort iff virtual ports are implemented.

If a RealPort has implemented virtual ports, the following information may be provided:

— maxVirtualPorts: The guaranteed maximum number of virtual ports.

— currentVirtualPorts: The current number of virtual ports.

12.9.3 Initialization

The following control allows a port to be reinitialized, terminating (and potentially restarting) authentication exchanges and MKA operation:

— initializePort(): The state machine variable BEGIN is asserted for each of the state machines defined in this standard, the MKA Live Peer and Potential Peer Lists are cleared, and all cached CAKs (but not PSKs) are deleted. If the port is a real port any virtual ports previously instantiated are deleted; if the port is a virtual port it is deleted, but can be reinstantiated through normal protocol action.

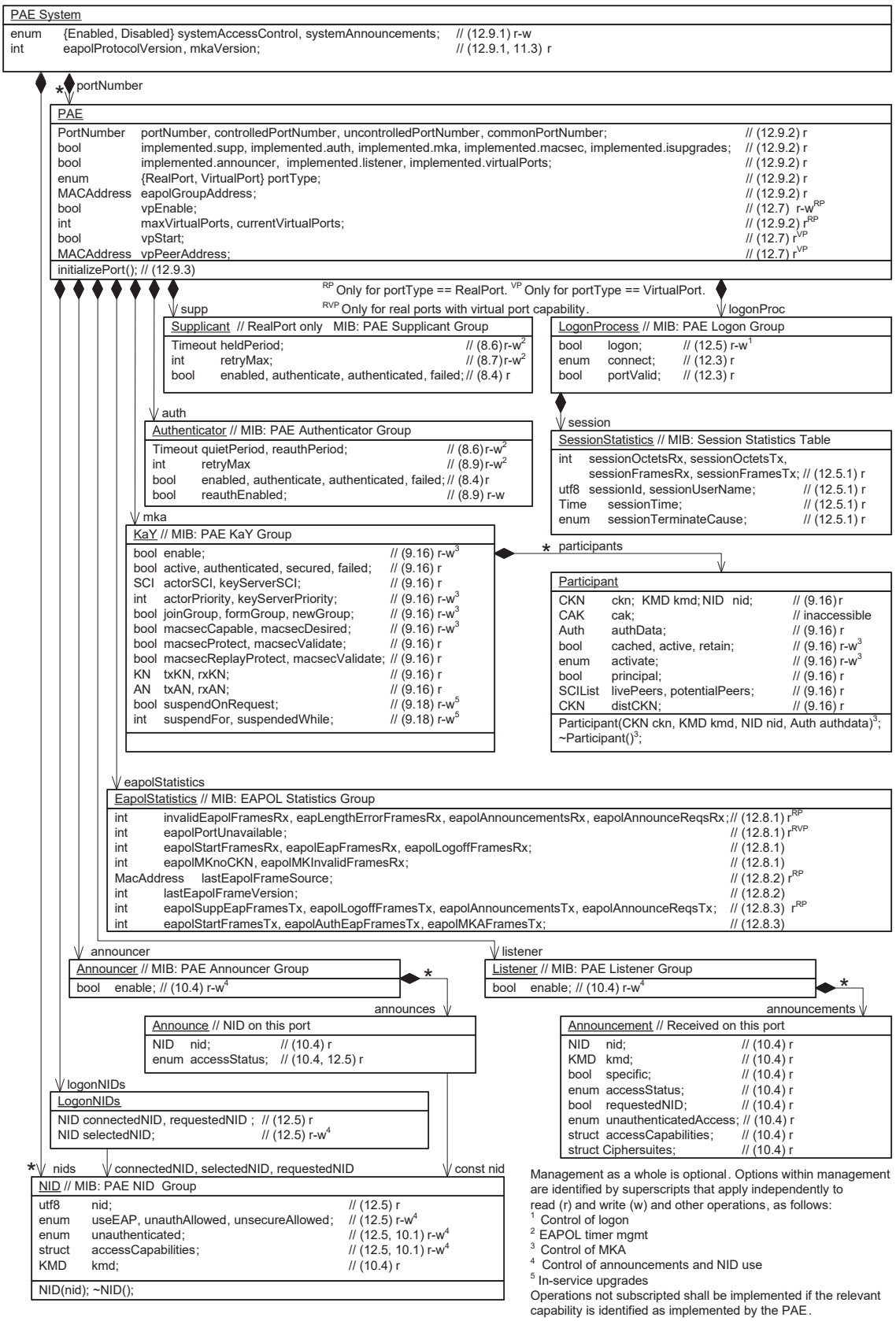


Figure 12-3—PAE management information

13. PAE MIB

This clause contains an SMIv2 Management Information Base (MIB) that facilitates control and monitoring of a system's PAEs' component protocol entities and processes, providing access to the operational controls, statistics and diagnostic capabilities specified in 12.9 (PAE management). The MIB also provides the managed objects necessary for a PAC if a SecY (see 6.1) is not used in conjunction with the PAE.

NOTE—IEEE Std 802.1X-2010 revised the MIB specified previously specified in IEEE Std 802.1X-2004 and used a distinct MIB root. The greater part of the present functionality was added (management of MKA, the Logon Process, virtual ports, and Announcements) and much of the previously specified functionality has been removed, including objects already deprecated (internal details of state machines, out of scope functionality e.g., EAP backend).

13.1 The Internet Standard Management Framework

For a detailed overview of the documents that describe the current Internet Standard Management Framework, please refer to Section 7 of IETF RFC 3410 [B9]. 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, IETF RFC 2578; IETF STD 58, IETF RFC 2579; and IETF STD 58, IETF RFC 2580.

13.2 Structure of the MIB

A single MIB module is defined in this clause. Objects in the MIB are arranged into groups. Table 13-4 and Table 13-5 provide cross-references for the PAE and PAC objects defined in this MIB.The overall structure and assignment of PAE objects to their groups follows that set out in Figure 12-3.

NOTE—Indentation of items in the left hand column of the tables reflects containment relationships, while '*' indicates a one to many reference as in Figure 12-3.

13.3 Relationship to other MIBs

13.3.1 System MIB Group

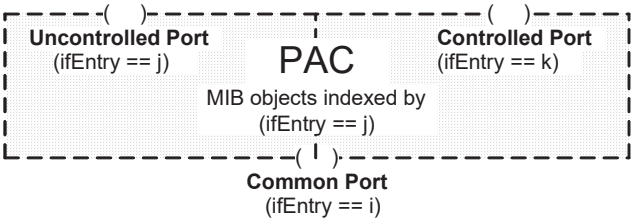
A system implementing this MIB shall also implement at least the System Group of the SNMPv2-MIB defined in IETF RFC 3418.

13.3.2 Relationship to the Interfaces MIB

A system implementing this MIB shall also implement the Interfaces Group (the Interfaces MIB module, or IF-MIB) defined in IETF RFC 2863. This MIB includes the clarifications mandated by IETF RFC 2863 for any MIB that is medium-specific or an adjunct of the Interfaces Group MIB.

The port numbers used in this standard are equal to the corresponding Port's ifIndex value.

The PAC managed objects specify attributes of a shim (IEEE Std 802.1AC) in an interface stack. The Controlled Port and Uncontrolled Ports are service access points provided by the PAC. Following IETF RFC 2863, these two service access points are defined as supported by separate sub-layers in the interface stack, and each has an individual conceptual row in the ifTable. The two interfaces are created together and co-exist without interference; one is not "on the top" of the other. The ifStackTable is used to identify the layer relationships between the (upper) Controlled Port interface provided by the PAC and the (lower) Controlled Port interface that it uses, and between the upper and lower Uncontrolled Ports (see Figure 13-1).



Higher Layer	Lower Layer
j	i
k	i

Figure 13-1—Use of the ifStackTable

The Controlled Port's interface type is macSecControlledIF(231), the Uncontrolled Port's interface type is macSecUncontrolledIF(232).

NOTE—These interface types were defined by IEEE Std 802.1AE-2006, initially for use with a SecY.

The MAC_Enabled and MAC_Operational parameters are mapped to the ifAdminStatus and ifOperStatus objects. The ifAdminStatus object for Controlled Port interface and Uncontrolled Port interface is read only. The attributes in Table 13-1, Table 13-2, and Table 13-3 are part of the required ifGeneralInformationGroup, ifCounterDiscontinuityGroup, and ifStackGroup2 object groups specified in IETF RFC 2863, and are not duplicated in the PAE MIB.

Table 13-1—Use of ifGeneralInformationGroup objects

ifGeneralInformationGroup objects	Use for MACsec
ifDescr	See interfaces MIB (IETF RFC 2863).
ifType	controlled port - macSecControlledIF(231) uncontrolled port - macSecUncontrolledIF(232)
ifSpeed	controlled port - same as the physical interface ifSpeed. uncontrolled port - same as the physical interface ifSpeed.
ifPhysAddress	This object should have an octet string with zero length for a KaY's controlled port and uncontrolled port.
ifAdminStatus	See interfaces MIB, read only for controlled port and uncontrolled port.
ifOperStatus	See interfaces MIB.
ifLastChange	See interfaces MIB.
ifName	See interfaces MIB.
ifLinkUpDownTrapEnable	See interfaces MIB, Default set as follows: controlled port - disabled(2) uncontrolled port - disabled(2)
ifHighSpeed	See interfaces MIB controlled port - same as the physical interfaces's ifHighSpeed. uncontrolled port - same as the physical interfaces's ifHighSpeed.
ifConnectorPresent	See interfaces MIB. Default set as follows: controlled port - false(2) uncontrolled port - false(2)
ifAlias	See interfaces MIB.
ifTableLastChange	See interfaces MIB.

Table 13-2—Use of ifCounterDiscontinuityGroup Object

ifCounterDiscontinuityGroup Object	Use for MACsec
ifCounterDiscontinuityTime	See interfaces MIB. controlled port: always 0, no discontinuity. uncontrolled port: always 0, no discontinuity.

Table 13-3—Use of ifStackGroup2 Objects

ifStackGroup2 Objects	Use for MACsec
ifStackStatus	See interfaces MIB.
ifStackLastChange	See interfaces MIB.

13.3.3 Relationship to the MAC Security MIB

The MIB module defined in this clause imports definitions from the IEEE8021-SECY-MIB specified in IEEE Std 802.1AE, and will be most often used in conjunction with that MIB. When MACsec is not supported on any port of a system, implementation of the IEEE8021-SECY-MIB for that system is not required.

Table 13-4—PAE managed object cross-reference table

PAE management information (Figure 12-3)	MIB object(s)
PAE System	**ieee8021XPaeSystem**
systemAccessControl (12.9.1) r–w	ieee8021XPaeSysAccessControl
systemAnnouncements (12.9.1) r–w	ieee8021XPaeSysAnnouncements
eapolProtocolVersion (12.9.1, 11.3) r	ieee8021XPaeSysEapolVersion
mkaVersion (12.9.1, 11.3) r	ieee8021XPaeSysMkaVersion

Table 13-4—PAE managed object cross-reference table (*continued*)

PAE management information (Figure 12-3)	MIB object(s)
*** PAE**	**ieee8021XPaePortTable** SEQUENCE OF **Ieee8021XPaePortEntry**
portNumber (12.9.2) r	ieee8021XPaePortNumber — INDEX
controlledPortNumber (12.9.2) r	ieee8021XPaeControlledPortNumber
uncontrolledPortNumber (12.9.2) r	ieee8021XPaeUncontrolledPortNumber
commonPortNumber (12.9.2) r	ieee8021XPaeCommonPortNumber
eapolGroupAddress (12.9.2) r	ieee8021XPaeEapolGroupMAC
initalizePort() (12.9.3)	ieee8021XPaePortInitialize
implemented (12.9.2) r	ieee8021XPaePortCapabilities
portType (12.9.2) r	ieee8021XPaePortType
vpEnable(12.7)	ieee8021XPaePortVirtualPortsEnable
maxVirtualPorts (12.9.2) r	ieee8021XPaePortMaxVirtualPorts
currentVirtualPorts (12.9.2) r	ieee8021XPaePortCurrentVirtualPorts
vpStart (12.7) r	ieee8021XPaePortVirtualPortStart
vpPeerAddress (12.7) r	ieee8021XPaePortVirtualPortPeerMAC
LogonProcess.logon (12.5) r	ieee8021XPaePortLogonEnable
Authenticator.enabled (8.4) r	ieee8021XPaePortAuthenticatorEnable
Supplicant.enabled (8.4) r	ieee8021XPaePortSupplicantEnable
KaY.enable (9.16) r-w	ieee8021XPaePortKayMkaEnable
Announcer.enable(10.4) r-w	ieee8021XPaePortAnnouncerEnable
Listener.enable (10.4) r-w	ieee8021XPaePortListenerEnable
LogonProcess	**ieee8021XPaePortLogonTable SEQUENCE OF Ieee8021XPaePortLogonEntry**
—	ieee8021XPaePortNumber -—INDEX
logon (12.5) r-w	— ieee8021XPaePortLogonEnable in **Ieee8021XPaePortEntry**
connect (12.3) r	ieee8021XPaePortLogonConnectStatus
portValid (12.3) r	ieee8021XPaePortPortValid
SessionStatistics	**ieee8021XPaePortSessionTable** SEQUENCE OF **Ieee8021XPaePortSessionEntry**
—	ieee8021XPaeControlledPortNumber — INDEX
sessionOctetsRx (12.5.4) r	ieee8021XPaePortSessionOctetsRx
sessionOctetsTx (12.5.4) r	ieee8021XPaePortSessionOctetsTx
sessionFramesRx (12.5.4) r	ieee8021XPaePortSessionPktsRx
sessionFramesTx (12.5.4) r	ieee8021XPaePortSessionPktsTx
sessionId (12.5.4) r	ieee8021XPaePortSessionId
—	ieee8021XPaePortSessionStartTime
sessionTime (12.5.4) r	ieee8021XPaePortSessionIntervalTime
sessionTerminateCause (12.5.4) r	ieee8021XPaePortSessionTerminate
sessionUserName (12.5.4) r	ieee8021XPaePortSessionUserName

Table 13-4—PAE managed object cross-reference table (*continued*)

PAE management information (Figure 12-3)	MIB object(s)
LogonNIDs	**ieee8021XLogonNIDTable** SEQUENCE OF **Ieee8021XLogonNIDEntry**
—	ieee8021XPaePortNumber — INDEX
connectedNID (12.5) r	ieee8021XLogonNIDConnectedNID
requestedNID (12.5) r	ieee8021XLogonNIDRequestedNID
selectedNID (12.5) r-w	ieee8021XLogonNIDSelectedNID
Authenticator	**ieee8021XAuthenticatorTable** SEQUENCE OF **Ieee8021XAuthenticatorEntry**
—	ieee8021XPaePortNumber — INDEX
enabled (8.4) r	— ieee8021XPaePortAuthenticatorEnable in **Ieee8021XPaePortEntry**
authenticate (8.4) r	ieee8021XAuthPaeAuthenticate
authenticated (8.4) r	ieee8021XAuthPaeAuthenticated
failed (8.4) r	ieee8021XAuthPaeFailed
reAuthEnabled (8.9) r-w	ieee8021XAuthPaeReAuthEnabled
quietPeriod (8.6) r-w	ieee8021XAuthPaeQuietPeriod
reauthPeriod (8.6)	ieee8021XAuthPaeReauthPeriod
retryMax (8.9) r-w	ieee8021XAuthPaeRetryMax
retryCount (8.9) r	ieee8021XAuthPaeRetryCount
Supplicant	**ieee8021XSupplicantTable** SEQUENCE OF **Ieee8021XSupplicantEntry**
—	ieee8021XPaePortNumber — INDEX
enabled (8.4) r	— ieee8021XPaePortSupplicantEnable in **Ieee8021XPaePortEntry**
authenticate (8.4) r	ieee8021XSuppPaeAuthenticate
authenticated (8.4) r	ieee8021XSuppPaeAuthenticated
failed (8.4) r	ieee8021XSuppPaeFailed
heldPeriod (8.6) r-w	ieee8021XSuppPaeHelloPeriod
retryMax (8.7) r-w	ieee8021XSuppPaeRetryMax
retryCount (8.7) r	ieee8021XSuppPaeRetryCount

Table 13-4—PAE managed object cross-reference table (*continued*)

PAE management information (Figure 12-3)	MIB object(s)
EAPOL	**ieee8021XEapolStatsTable** SEQUENCE OF **Ieee8021XEapolStatsEntry**
—	ieee8021XPaePortNumber — INDEX
invalidEapolFramesRx (12.8.1) r	ieee8021XEapolInvalidFramesRx
eapLengthErrorFrames (12.8.1) r	ieee8021XEapolEapLengthErrorFramesRx
eapolAnnouncementsRx (12.8.1) r	ieee8021XEapolFramesAnnouncementFramesRx
eapolAnnounceReqsRx (12.8.1) r	ieee8021XEapolAnnouncementReqFramesRx
eapolPortUnavailable (12.8.1) r	ieee8021XEapolPortUnavailableFramesRx
eapolStartFramesRx (12.8.1) r	ieee8021XEapolStartFramesRx
eapolEapFramesRx (12.8.1) r	ieee8021XEapolEapFramesRx
eapolLogoffFramesRx (12.8.1) r	ieee8021XEapolLogoffFramesRx
eapolMKnoCKN (12.8.1)	ieee8021XEapolMkNoCknFramesRx
eapolMKInvalidFramesRx (12.8.1)	ieee8021XEapolMkInvalidFramesRx
lastEAPOLFrameVersion (12.8.2) r	ieee8021XEapolLastRxFrameVersion
lastEAPOLFrameSource (12.8.2) r	ieee8021XEapolLastEapolFrameSource
eapolSuppEapFramesTx (12.8.3) r	ieee8021XEapolSuppEapFramesTx
eapolLogoffFramesTx (12.8.3) r	ieee8021XEapolLogoffFramesTx
eapolAnnouncementsTx (12.8.3) r	ieee8021XEapolAnnouncementFramesTx
eapolAnnounceReqsTx (12.8.3) r	ieee8021XEapolAnnouncementReqFramesTx
eapolStartFramesTx (12.8.3) r	ieee8021XEapolStartFramesTx
eapolAuthEapFramesTx (12.8.3) r	ieee8021XEapolAuthEapFramesTx
eapolMKAFramesTx (12.8.3) r	ieee8021XEapolMkaFramesTx

Table 13-4—PAE managed object cross-reference table (*continued*)

PAE management information (Figure 12-3)	MIB object(s)
KaY	**ieee8021XKayMkaTable** SEQUENCE OF **Ieee8021XKayMkaEntry**
—	ieee8021XPaePortNumber — INDEX
enable (9.16) r-w	— ieee8021XPaePortKayMkaEnable in **Ieee8021XPaePortEntry**
active (9.16) r	ieee8021XKayMkaActive
authenticated (9.16) r	ieee8021XKayMkaAuthenticated
secured (9.16) r	ieee8021XKayMkaSecured
failed (9.16) r	ieee8021XKayMkaFailed
actorSCI (9.16) r	ieee8021XKayMkaActorSCI
actorPriority (9.16) r-w	ieee8021XKayMkaActorsPriority
keyServerPriority (9.16) r-w	ieee8021XKayMkaKeyServerPriority
keyServerSCI (9.16) r	ieee8021XKayMkaKeyServerSCI
joinGroup (9.16) r-w	ieee8021XKayAllowedJoinGroup
formGroup (9.16) r-w	ieee8021XKayAllowedFormGroup
newGroup (9.16) r-w	ieee8021XKayCreateNewGroup
macsecCapable (9.16) r-w	ieee8021XKayMacSecCapability
macsecDesired (9.16) r-w	ieee8021XKayMacSecDesired
macsecProtect (9.16) r	ieee8021XKayMacSecProtect
macsecReplayProtect (9.16) r	ieee8021XKayMacSecReplayProtect
macsecValidate (9.16) r	ieee8021XKayMacSecValidate
macsecConfidentiality (9.16) r-w	ieee8021XKayMacSecConfidentialityOffset
txKN (9.16) r	ieee8021XKayMkaTxKN
txAN (9.16) r	ieee8021XKayMkaTxAN
rxKN (9.16) r	ieee8021XKayMkaRxKN
rxAN (9.16) r	ieee8021XKayMkaRxAN
suspendFor (9.16) r-w	ieee8021XKayMkaSuspendFor
suspendOnRequest (9.16) r-w	ieee8021XKayMkaSuspendOnRequest
suspendWhile (9.16) r-w	ieee8021XKayMkaSuspendedWhile

Table 13-4—PAE managed object cross-reference table (*continued*)

PAE management information (Figure 12-3)	MIB object(s)
* **Participant**	**ieee8021XKayMKAParticipantTable** SEQUENCE OF **Ieee8021XKayMkaParticipantEntry**
—	ieee8021XPaePortNumber — INDEX
ckn (9.16) r	ieee8021XKayMkaPartCKN—INDEX (IMPLIED)
kmd (9.16) r	ieee8021XKayMkaPartKMD
nid (9.16) r	ieee8021XKayMkaPartNID
authData (9.16) r	— not MIB accessible
cached (9.16) r-w	ieee8021XKayMkaPartCached
active (9.16) r-w	ieee8021XKayMkaPartActive
retain (9.16) r-w	ieee8021XKayMkaPartRetain
activate (9.16) r-w	ieee8021XKayMkaPartActiveControl
principal (9.16) r	ieee8021XKayMkaPartPrincipal
distCKN (9.16) r	ieee8021XKayMkaPartDistCKN
Participant(..);~Participant();	ieee8021XKayMkaPartRowStatus
Participant.livePeers **Participant.potentialPeers** (for all Participants)	**ieee8021XKayMKAPeerListTable** SEQUENCE OF **Ieee8021XKayMkaPeerListEntry**
—	ieee8021XPaePortNumber — INDEX
—	ieee8021XKayMkaPartCKN—INDEX
—	ieee8021XKayMkaPeerListMI—INDEX
—	ieee8021XKayMkaPeerListMN
SCIList (one entry) (9.16) r	ieee8021XKayMkaPeerListSCI

Table 13-4—PAE managed object cross-reference table (*continued*)

PAE management information (Figure 12-3)	MIB object(s)
*** NID**	**ieee8021XNidConfigTable** SEQUENCE OF **Ieee8021XNIDConfigEntry**
nid (12.5) r	ieee8021XNidNid — INDEX (IMPLIED)
useEAP (12.5) r-w	ieee8021XNidUseEap
unauthAllowed (12.5) r-w	ieee8021XNidUnauthAllowed
unsecureAllowed (12.5) r-w	ieee8021XNidUnsecuredAllowed
unauthenticated (12.5, 10.1) r-w	ieee8021XNidUnauthenticatedAccess
accessCapabilities (12.5, 10.1) r-w	ieee8021XNidAccessCapabilities
kmd (10.4) r	ieee8021XNidKMD
NID(nid); ~NID();	ieee8021XNidRowStatus
*** PAE.Announcer**	—
enable	— ieee8021XPaePortAnnouncerEnable in **Ieee8021XPaePortEntry**
* Announce	ieee8021XAnnounceTable SEQUENCE OF AnnounceEntry
—	ieee8021XPaePortNumber — INDEX
nid (10.4) r	ieee8021XAnnounceNID
accessStatus (10.4, 12.5) r	ieee8021XAnnounceAccessStatus
*** PAE.Listener**	—
enable	— ieee8021XPaePortListenerEnable in **Ieee8021XPaePortEntry**
* Announcement	ieee8021XAnnouncementTable SEQUENCE OF **Ieee8021XAnnouncementEntry**
—	ieee8021XPaePortNumber — INDEX
nid (10.4) r	ieee8021XAnnouncementNID — INDEX (IMPLIED)
kmd (10.4) r	ieee8021XAnnouncementKMD
specific (10.4) r	ieee8021XAnnouncementSpecific
accessStatus (10.4) r	ieee8021XAnnouncementAccessStatus
requestedNID (10.4) r	ieee8021XAnnouncementAccessRequested
unauthenticatedAccess (10.4) r	ieee8021XAnnouncementUnauthAccess
accessCapabilities (10.4) r	ieee8021XAnnouncementCapabilities
Ciphersuites (10.4) r	ieee8021XAnnouncementCipherSuitesTable

Table 13-5—PAC managed object cross-reference table

PAC management information	MIB object(s)
*** PAC**	**ieee8021XPacPortTable** SEQUENCE OF **Ieee8021XPacPortEntry**
portNumber	ieee8021XPaePortNumber — INDEX
adminPointToPointMAC (6.4.3)	ieee8021XPacPortAdminPt2PtMac
operPointToPointMAC (6.4.3)	ieee8021XPacPortOperPt2PtMac

13.4 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 (see 5.22). Instead, implementors can deploy SNMPv3 to enable cryptographic security. It is then the responsibility of a system's administrator to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

This standard does not specify remote read access to certain variables, such as secret keys, that are stored and used without being disclosed directly and refrains from defining any management operations that read those variables. Proprietary extensions to this MIB, or to others in the system, that could provide such access are likely to compromise the security provided and could negate the purpose of this standard. Implementations of this standard should consider mechanisms to ensure such keys can only be used for specified operations (such as calculating an ICV).

All management objects defined in this MIB with a MAX-ACCESS of other than not-accessible can be considered sensitive or vulnerable in some network environments, with inappropriate access having a negative effect on network operations. It is thus important to control even GET and/or NOTIFY access to these objects, and encryption of values sent over the network via SNMP should be the default in the absence of detailed analysis.

The management objects defined in this MIB with a MAX-ACCESS clause of read-write or read-create that are most likely to be considered sensitive or vulnerable include the following. One or more ways in which an attacker could compromise the network or a system by writing or creating each object are noted, to illustrate the vulnerability, but this clause does not attempt to list the full range of possible attacks.

a) systemAccessControl: Turn off network access control completely, thus rendering a network or a system accessing a network open to attack.
b) ieee8021XPaePortKayMkaEnable: Turn off MKA operation, thus preventing use of MACsec, either denying service or rendering communication open to attack.
c) nidConfigTable: Modify policies for unsecured or unauthenticated communication, rendering the system open to attack or denying service (and thus encouraging security to be disabled).

The readable management objects defined in this MIB (i.e., objects with a MAX-ACCESS other than not-accessible) that are most likely to be considered sensitive or vulnerable include the following. One or more ways in which an attacker could make use of the information to compromise a network or the system are noted, but this clause does not attempt to list the full range of possible attack.

d) systemAccessControl: Identify network access points that are not configured to secure access.
e) nidConfigTable: Identify network access points that permit unauthenticated or unsecured access to certain network services.

NOTE—IEEE 802.1Xbx-2014 added the in-service upgrade (9.18) group (ieee8021XPaeKaYIsupgradeGroup) to this standard. IEEE 802.1Xck-2018 added the EAPOL Group Address used by each PAE (12.9.2) (ieee8021XPaeEapolGroupMAC).These additions do not affect the security considerations to be taken into account when making use of this standard.

13.5 Definitions for PAE MIB[22]

In the MIB definition below, should there be any discrepancy between the DESCRIPTION text and the other clauses of this standard, the latter shall take precedence.

[22] *Copyright release for MIBs:* Users of this standard may freely reproduce the MIB definition contained in this clause so that it can be used for its intended purpose.

```
-- ****************************************************************
--
-- IEEE8021X-PAE-MIB : IEEE 802.1X
--
-- ****************************************************************

IEEE8021X-PAE-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Gauge32,
    Counter32,
    Counter64,
    Unsigned32,
    Integer32
        FROM SNMPv2-SMI
    MacAddress,
    TEXTUAL-CONVENTION,
    TruthValue,
    RowPointer,
    TimeStamp,
    TimeInterval,
    RowStatus
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    InterfaceIndex
        FROM IF-MIB
    SecySCI
        FROM IEEE8021-SECY-MIB;

ieee8021XPaeMIB MODULE-IDENTITY
    LAST-UPDATED    "202002181507Z"
    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@LISTSERV.IEEE.ORG"
    DESCRIPTION
        "The MIB module for managing the Port Access Entity (PAE)
        functions of IEEE 802.1X (Revision of 802.1X-2004).
        The PAE functions managed are summarized in Figure 12-3 of
        IEEE 802.1X and include EAPOL PACP support for authentication
        (EAP Supplicant and/or Authenticator), MACsec Key Agreement
        (MKA), EAPOL, and transmission and reception of network
        announcements.

        The following acronyms and definitions are used in this MIB.

        AN : Association Number, a number that is concatenated with a
            MACsec Secure Channel Identifier to identify a Secure
            Association (SA).

        Announcer : EAPOL-Announcement transmission functionality.

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

        CA : secure Connectivity Association: A security relationship,
            established and maintained by key agreement protocols, that
            comprises a fully connected subset of the service access
```

points in stations attached to a single LAN that are to be supported by MACsec.

CAK : secure Connectivity Association Key, a secret key possessed by members of a given CA.

CKN : secure Connectivity Association Key Name (CKN), a text that identifies a CAK.

Common Port : An instance of the MAC Internal Sublayer Service used by the SecY or PAC to provide transmission and reception of frames for both the Controlled and Uncontrolled Ports.

Controlled Port : The access point used to provide the secure MAC Service to a client of a PAC or SecY.

CP state machine : Controlled Port state machine is capable of controlling a SecY or a PAC. The CP supports interoperability with unauthenticated systems that are not port-based network access control capable, or that lack MKA. When the access controlled port is supported by a SecY, the CP is capable of controlling the SecY so as to provide unsecured connectivity to systems that implement a PAC.

EAP : Extensible Authentication Protocol, RFC3748.

EAPOL : EAP over LANs.

KaY : Key Agreement Entity, a PAE entity responsible for MKA.

Key Server : Elected by MKA, to transport a succession of SAKs, for use by MACsec, to the other member(s) of a CA.

KMD : Key Management Domain, a string identifying systems that share cached CAKs.

Listener : The role is to receive the network announcement parameters in the authentication process.

Logon Process : The Logon Process is responsible for the managing the use of authentication credentials, for initiating use of the PAE’s Supplicant and or Authenticator functionality, for deriving CAK, CKN tuples from PAE results, for maintaining PSKs (Pre-Sharing Keys), and for managing MKA instances. In the absence of successful authentication, key agreement, or support for MAC Security, the Logon Process determines whether the CP state machine should provide unauthenticated connectivity or authenticated but unsecured connectivity.

MKA : MACsec Key Agreement protocol allows PAEs, each associated with a port that is an authenticated member of a secure connectivity association (CA) or a potential CA, to discover other PAEs attached to the same LAN, to confirm mutual possession of a CAK and hence to prove a past mutual authentication, to agree the secret keys (SAKs) used by MACsec for symmetric shared key cryptography, and to ensure that the data protected by MACsec has not been delayed.

MKPDU : MACsec Key Agreement Protocol Data Unit.

MPDU : MAC Protocol Data Unit.

NID : Network Identity, a UTF-8 string identifying an network or network service.

PAE : Port Access Entity, the protocol entity associated with a Port. It can support the protocol functionality associated with the Authenticator, the Supplicant, or both.

PAC : Port Access Controller, a protocol-less shim that provides control over frame transmission and reception by clients attached to its Controlled Port, and uses the MAC Service provided by a Common Port. The access control decision is made by the PAE, typically taking into account the success or failure of mutual authentication and authorization of the PAE's peer(s), and is communicated by the PAE using the LMI to set the PAC's Controlled Port enabled/disable. Two different interfaces 'Controlled Port' and 'Uncontrolled Port', are associated with a PAC, and that for each instance of a PAC, two ifTable rows (one for each interface) run on top of an ifTable row representing the 'Common Port' interface, such as a row with ifType = 'ethernetCsmacd(6)'.

For example :

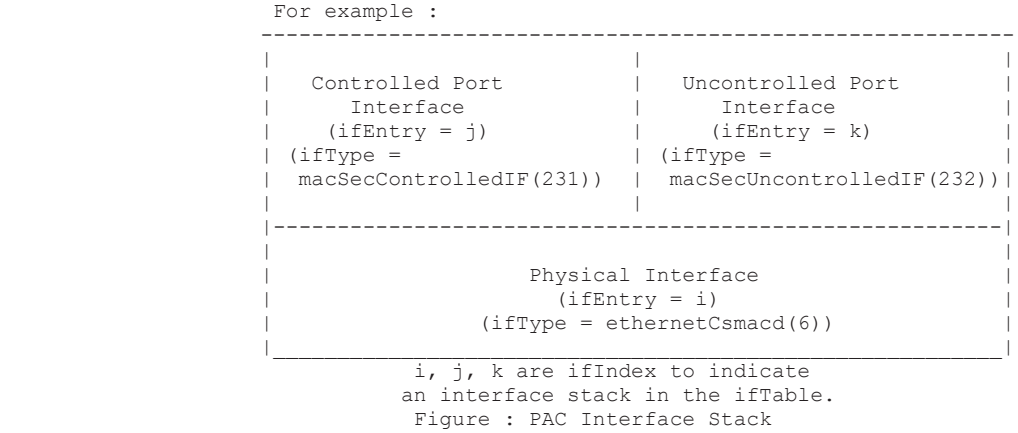


Figure : PAC Interface Stack

The 'Controlled Port' is the service point to provide one instance of the secure MAC service in a PAC. The 'Uncontrolled Port' is the service point to provide one instance of the insecure MAC service in a PAC.

PACP : Port Access Controller Protocol.

Port Identifier : A 16-bit identifier that uniquely identifies each of a system's transmit SCs that uses the same MAC address as a component of its SCI.

Real Port : Indicates the PAE is for a real port. A port that is not created on demand by the mechanisms specified in this standard, but that can transmit and receive frames for one or more virtual ports.

SC : Secure Channel, a security relationship used to provide security guarantees for frames transmitted from one member of a CA to the others. An SC is supported by a sequence of SAs thus allowing the periodic use of fresh keys without terminating the relationship.

SA : Secure Association, a security relationship that provides security guarantees for frames transmitted from one member of a CA to the others. Each SA is supported by a single secret key, or a single set of keys where the cryptographic operations used to protect one frame require more than one key.

SAK : Secure Association key, the secret key used by an SA.

SCI : Secure Channel Identifier, a unique identifier for a secure channel, comprising a MAC Address and a Port Identifier.

secured connectivity : Data transfer between two or 'Controlled Ports' that is protected by MACsec.

```
        SecY : MAC Security Entity, the entity that operates the MAC
            Security protocol within a system.

        Supplicant : An entity at one end of a point-to-point LAN
            segment that seeks to be authenticated by an Authenticator
            attached to the other end of that link.

        Suspension: Temporary suspension of MKA operation to facilitate
            in-service control plane software upgrades without
            disrupting existing secure connectivity.

        Uncontrolled Port : The access point used to provide the
            insecure MAC Service to a client of a SecY or PAC.

        Virtual Port : Indicates the PAE is for a virtual port.  A MAC
            Service or Internal Sublayer service access point that is
            created on demand.  Virtual ports can be used to provide
            separate secure connectivity associations over the same
            LAN."
    REVISION        "201202181507Z"
    DESCRIPTION
        "Published as part of IEEE Std 802.1X-2020.
         Updated CONTACT-INFO. Corrected last REVISION DESCRIPTION."
    REVISION        "2019041020402"
    DESCRIPTION
        "Edited ieee8021XSuppPaeHelloPeriod DESCRIPTION to refer to HeldPeriod.
         Updated ieee8021XSuppPaeRetryMax DESCRIPTION to match."
    REVISION        "201710281457Z"
    DESCRIPTION
        "Published as part of IEEE 802.1Xck.
         Minor DESCRIPTION clarifications as required by resolution
         of maintenance items 154, 155, 157 (see 802.1 maintenance
         process discussion). Added ieee8021XPaeEapolGroupMAC Address."
    REVISION        "201404101619Z"
    DESCRIPTION
        "Update published as part of IEEE 802.1Xbx (Amendment to
        IEEE 802.1X-2010)"
    REVISION        "200910011650Z"
    DESCRIPTION
        "Initial version of this MIB module.  Published as part of
        IEEE P802.1X (Revision of IEEE Standard 802.1X-2009)"
    ::= { iso(1) iso-identified-organization(3) ieee(111)
          standards-association-numbered-series-standards(2)
          lan-man-stds(802) ieee802dot1(1) ieee802dot1mibs(1) 15 }

-- ------------------------------------------------------------------ --
-- Textual Conventions
-- ------------------------------------------------------------------ --

Ieee8021XPaeCKN ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates the CAK name to identify
        the Connectivity Association Key (CAK) which is the root key
        in the MACsec Key Agreement key hierarchy.  All potential
        members of the CA use the same CKN."

    REFERENCE       "IEEE 802.1X Clause 5.4, Clause 9.3.1, Clause 6.2"
    SYNTAX          OCTET STRING (SIZE (1..16))

Ieee8021XPaeCKNOrNull ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates the CAK name to identify
        the Connectivity Association Key (CAK) which is the root key
        in the MACsec Key Agreement key hierarchy.  All potential
        members of the CA use the same CKN.

        If this is a zero length value, then the NULL string means
        CKN information is applicable."

    REFERENCE       "IEEE 802.1X Clause 5.4, Clause 9.3.1, Clause 6.2"
```

```
    SYNTAX          OCTET STRING (SIZE (0..16))

Ieee8021XPaeKMD ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Key Management Domain
        (KMD).

        KMD is a string of UTF-8 characters that names the transmitting
        authenticator's key management domain."

    REFERENCE       "IEEE 802.1X Clause 12.6"
    SYNTAX          OCTET STRING (SIZE (0..253))

Ieee8021XPaeNID ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Network Identifier (NID).

        Each network is identified by a NID, a UTF-8 string used by
        network attached systems to select a network profile."

    REFERENCE       "IEEE 802.1X Clause 12.6, Clause 10.1"
    SYNTAX          OCTET STRING (SIZE (1..100))

Ieee8021XPaeNIDOrNull ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Network Identifier (NID).

        Each network is identified by a NID, a UTF-8 string used by
        network attached systems to select a network profile.

        If this is a zero length value, then the NULL string for
        NID information is applicable."

    REFERENCE       "IEEE 802.1X Clause 12.6, Clause 10.1"
    SYNTAX          OCTET STRING (SIZE (0..100))

Ieee8021XMkaKeyServerPriority ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Key Server priority
        information.

        Each MKA participant encodes a Key Server Priority, an 8-bit
        integer, in each MKPDU.  Each participant selects the live
        participant advertising the highest priority as its Key Server
        provided that participant has not selected another as its Key
        Server or is unwilling to act as the Key Server.  If a Key
        Server cannot be selected SAKs are not distributed.  In the
        event of a tie for highest priority Key Server, the member with
        the highest priority SCI is chosen.  For consistency with other
        uses of the SCI's MAC Address component as a priority,
        numerically lower values of the Key Server Priority and SCI are
        accorded the highest priority.  The Table 9-2 contains
        recommendations for the use of priority values for various
        system roles. Participants that will never act as a Key Server
        should advertise priority 0xFF."

    REFERENCE       "IEEE 802.1X Clause 9.5, Table 9-2"
    SYNTAX          OCTET STRING (SIZE (1))

Ieee8021XMkaMI ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Member Identifier (MI).

        The MI is a 96-bit random value chosen when the MKA Instance
        begins, used with a 32-bit MN to protect against replay attacks
        and to record liveliness in the Live Peer List or potential
        liveliness in the Potential Peer List. If the MN wraps, a new
```

```
        random MI value is chosen and the MN begins again at 1."

    REFERENCE       "IEEE 802.1X Clause 9.4.2"
    SYNTAX          OCTET STRING (SIZE (12))

Ieee8021XMkaMN ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Member Number (MN).

        The MN is a 32-bit value which begins at 1 and increases for
        each MKPDU transmitted.  It is used with the MI to protect
        against replay attacks and to record liveliness in the Live
        Peers List or potential liveliness in the Potential Peer List.
        If the MN wraps, a new random MI value is chosen and the MN
        begins again at a value of 1."

    REFERENCE       "IEEE 802.1X Clause 9.4.2"
    SYNTAX          Unsigned32 (1..2147483648)

Ieee8021XMkaKN ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "This textual convention indicates a Key Number (KN) used in
        MKA.

        The MN is a 32-bit integer assigned by that Key Server
        (sequentially, beginning with 1)."

    REFERENCE       "IEEE 802.1X Clause 9.8"
    SYNTAX          Unsigned32 (1..2147483648)

Ieee8021XPaeNIDCapabilites ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates the combinations of
        authentication and protection capabilities supported for a
        NID. Any set of these combinations can be supported."

    REFERENCE       "IEEE 802.1X Clause 10.1, Table 11-8"
    SYNTAX          BITS {
                        eap(0),
                        eapMka(1),
                        eapMkaMacSec(2),
                        mka(3),
                        mkaMacSec(4),
                        higherLayer(5), -- WebAuth
                        higherLayerFallback(6), -- WebAuth
                        vendorSpecific(7)
                    }

Ieee8021XPaeNIDAccessStatus ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates the transmitter's
        Controlled Port operational status and current level of
        access resulting from authentication and the consequent
        authorization controls applied by that port's clients.

        'noAccess' : Other than to authentication services, and to
            services announced as available in the absence of
            authentication (unauthenticated).

        'remedialAccess' : The access granted is severely limited,
            possibly to remedial services.

        'restrictedAccess' : The Controlled Port is operational, but
            restrictions have been applied by the network that can
            limit access to some resources.
```

```
        'expectedAccess' : The Controlled Port is operational, and
            access provided is as expected for successful
            authentication and authorization for the NID."

    REFERENCE       "IEEE 802.1X Clause 10.1, Table 11-8"
    SYNTAX          INTEGER  {
                        noAccess(0),
                        remedialAccess(1),
                        restrictedAccess(2),
                        expectedAccess(3)
                    }

Ieee8021XPaeNIDUnauthenticatedStatus ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This textual convention indicates the access capabilities of
        the port's clients without authentication.

        'noAccess' : Other than to authentication services (see
            Ieee8021XPaeNIDCapabilites information.

        'fallbackAccess' : Limited access can be provided after
            authentication failure.

        'limitedAccess' : Immediate limited access is available
            without authentication.

        'openAccess' : Immediate access is available without
            authentication."

    REFERENCE       "IEEE 802.1X Clause 10.1, Table 11-8"
    SYNTAX          INTEGER  {
                        noAccess(0),
                        fallbackAccess(1),
                        limitedAccess(2),
                        openAccess(3)
                    }

-- ------------------------------------------------------------------ --
-- Groups in the IEEE 802.1X MIB
-- ------------------------------------------------------------------ --

ieee8021XPaeMIBNotifications  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIB 0 }

ieee8021XPaeMIBObjects  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIB 1 }

ieee8021XPaeMIBConformance  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIB 2 }

-- ------------------------------------------------------------------ --
-- Management Objects in the IEEE 802.1X MIB
-- ------------------------------------------------------------------ --

ieee8021XPaeSystem  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 1 }

ieee8021XPaeLogon  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 2 }

ieee8021XPaeAuthenticator  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 3 }

ieee8021XPaeSupplicant  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 4 }

ieee8021XPaeEapol  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 5 }

ieee8021XPaeKaY  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 6 }
```

```
ieee8021XPaeNetworkIdentifier  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBObjects 7 }


-- ------------------------------------------------------------------ --
-- The 802.1X PAE System Group
-- ------------------------------------------------------------------ --
--
-- ------------------------------------------------------------------ --
-- The 802.1X PAE System Objects
-- ------------------------------------------------------------------ --

ieee8021XPaeSysAccessControl OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "This object enables or disables port-based network access
        control for all the system's ports.  Setting this control
        object to 'false' causes the following actions :
            . Deletes any virtual ports previously instantiated.
            . Terminates authentication exchanges and MKA instances'
              operation.
            . Each real port PAE behaves as if no virtual ports
              created.
            . All the PAEs' Supplicant, Authenticator, and KaY are
              disabled.
            . Logon Process(es) behave as if the object
              ieee8021XNidUnauthAllowed was 'immediate'.
            . Announcements can be transmitted, both periodically and
              in response to announcement requests (conveyed by
              EAPOL-Starts or EAPOL-Announcement-Reqs) but are sent
              with a single NULL NID.
            . Objects announcementAccessStatus and announceAccessStatus
              have the 'noAccess' value, announcementAccessRequested is
              'false', object announcementUnauthAccess has the
              'openAccess' value.

        The control variable settings for each real port PAE in the
        ieee8021XPaePortTable are unaffected, and will be used once the
        object is set to 'true'.

        This configured value for this object shall be stored in
        persistent memory and remain unchanged across a
        re-initialization of the management system of the entity."
    REFERENCE
        "IEEE 802.1X Clause 12.9.1, Figure 12-3 PAE
            System.systemAccessControl"
    ::= { ieee8021XPaeSystem 1 }

ieee8021XPaeSysAnnouncements OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "Setting this control object to 'false' causes each PAE in this
        system to behave as if the PAE's Announcement functionality is
        disabled.  The independent controls for each PAE apply if
        this object is 'true'.

        This configured value for this object shall be stored in
        persistent memory and remain unchanged across a
        re-initialization of the management system of the entity."
    REFERENCE
        "IEEE 802.1X Clause 12.9.1, Figure 12-3 PAE
             System.systemAnnouncements"
    ::= { ieee8021XPaeSystem 2 }

ieee8021XPaeSysEapolVersion OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
```

```
    STATUS          current
    DESCRIPTION
        "The EAPOL protocol version for this system."
    REFERENCE
        "IEEE 802.1X Clause 12.9.1, Clause 11.3, Figure 12-3 PAE
             System.eapolProtocolVersion"
    ::= { ieee8021XPaeSystem 3 }

ieee8021XPaeSysMkaVersion OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The MKA protocol version for this system."
    REFERENCE       "IEEE 802.1X Clause 12.9.1"
    ::= { ieee8021XPaeSystem 4 }
-- ------------------------------------------------------------------ --
-- The 802.1X PAE Port Table
-- ------------------------------------------------------------------ --

ieee8021XPaePortTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XPaePortEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table of system level information for each port supported by
        the Port Access Entity.  An entry appears in this table for
        each port of this system.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3 PAE"
    ::= { ieee8021XPaeSystem 5 }

ieee8021XPaePortEntry OBJECT-TYPE
    SYNTAX          Ieee8021XPaePortEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The Port number, protocol version, and
        initialization control for a Port.

         If the PAE has been dynamically instantiated to support an
         existing or potential virtual port, the Uncontrolled Port
         interface and Controlled Port interface are allocated by the
         real port's PAE."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XPaePortTable 1 }

Ieee8021XPaePortEntry ::= SEQUENCE {
        ieee8021XPaePortNumber              InterfaceIndex,
        ieee8021XPaePortType                INTEGER,
        ieee8021XPaeControlledPortNumber    InterfaceIndex,
        ieee8021XPaeUncontrolledPortNumber  InterfaceIndex,
        ieee8021XPaeCommonPortNumber        InterfaceIndex,
        ieee8021XPaePortInitialize          TruthValue,
        ieee8021XPaePortCapabilities        BITS,
        ieee8021XPaePortVirtualPortsEnable  TruthValue,
        ieee8021XPaePortMaxVirtualPorts     Unsigned32,
        ieee8021XPaePortCurrentVirtualPorts Gauge32,
        ieee8021XPaePortVirtualPortStart    TruthValue,
        ieee8021XPaePortVirtualPortPeerMAC  MacAddress,
        ieee8021XPaePortLogonEnable         TruthValue,
        ieee8021XPaePortAuthenticatorEnable TruthValue,
        ieee8021XPaePortSupplicantEnable    TruthValue,
        ieee8021XPaePortKayMkaEnable        TruthValue,
        ieee8021XPaePortAnnouncerEnable     TruthValue,
        ieee8021XPaePortListenerEnable      TruthValue,
        ieee8021XPaeEapolGroupMACMacAddress
}
```

```
ieee8021XPaePortNumber OBJECT-TYPE
    SYNTAX          InterfaceIndex
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An interface index indicates the port number associated with
        this port.  Each PAE is uniquely identified by a port number.
        The port number used is unique amongst all port numbers for
        the system, and directly or indirectly identifies the
        Uncontrolled Port that supports the PAE.

        If the PAE indicates a real port, ieee8021XPaePortType object
        in the same row is 'realPort', the port number shall be the
        same as the ieee8021XPaeCommonPortNumber object in the same row
        for the associated PAC or SecY.

        If the PAE indicates a virtual port, ieee8021XPaePortType
        object in the same row is 'virtualPort', this port number
        should be the same as the uncontrolledPortNumber object in the
        same row for the associated PAC or SecY."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 1 }

ieee8021XPaePortType OBJECT-TYPE
    SYNTAX          INTEGER  {
                        realPort(1),
                        virtualPort(2)
                    }
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The port type of the PAE.

        realPort(1) : indicates the PAE is for a real port.

        virtualPort(2) : indicates the PAE is for a virtual port."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 2 }

ieee8021XPaeControlledPortNumber OBJECT-TYPE
    SYNTAX          InterfaceIndex
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "An interface index indicates the port number associated with
        PAC or SecY's Controlled Port."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 3 }

ieee8021XPaeUncontrolledPortNumber OBJECT-TYPE
    SYNTAX          InterfaceIndex
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "An interface index indicates the port number associated with
        PAC or SecY's Uncontrolled Port.  If the PAE supports a
        real port, this port number can be the same as the
        ieee8021XPaeCommonPortNumber object in the same row, otherwise
        it shall not be the same."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 4 }

ieee8021XPaeCommonPortNumber OBJECT-TYPE
    SYNTAX          InterfaceIndex
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "An interface index indicates the port number associated with
        PAC or SecY's 'Common Port'.  All the virtual ports created
        for a given real port share the same 'Common Port' and
        ieee8021XPaeCommonPortNumber in the same row."
```

```
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 5 }

ieee8021XPaePortInitialize OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The initialization control for this Port. Setting this object
        'true' causes the Port to be reinitialized, terminating (and
        potentially restarting) authentication exchanges and MKA
        operation.

        If the port is a real port, any virtual ports previously
        instantiated are deleted.  Virtual ports can be reinstantiated
        through normal protocol operation.

        The object value reverts to 'false' once initialization
        has completed."
    REFERENCE       "802.1X Clause 12.9.3, Figure 12-3"
    ::= { ieee8021XPaePortEntry 6 }

ieee8021XPaePortCapabilities OBJECT-TYPE
    SYNTAX          BITS {
                        suppImplemented(0),
                        authImplemented(1),
                        mkaImplemented(2),
                        macsecImplemented(3),
                        announcementsImplemented(4),
                        listenerImplemented(5),
                        virtualPortsImplemented(6)
                    }
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The capabilities of this PAE port.

        'suppImplemented' : A PACP EAP supplicant functions are
            implemented in this PAE if this bit is on.

        'authImplemented' : A PACP EAP authenticator functions are
            implemented in this PAE if this bit is on.

        'mkaImplemented' : The KaY MKA functions are implemented
            in this PAE if this bit is on.

        'macsecImplemented' : The MACsec functions in the
            Controlled Port are implemented in this PAE if this
            bit is on.

        'announcementsImplemented' : The EAPOL announcement can be
            sent in this PAE if this bit is on.

        'listenerImplemented' : This PAE can receive EAPOL announcement
            if this bit is on.

        'virtualPortsImplemented' : Virtual Port functions are
            implemented in this PAE if this bit is on."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 7 }

ieee8021XPaePortVirtualPortsEnable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "Enable or disable to Virtual Ports function for this Real Port
        PAE, the object ieee8021XPaePortType in the same row has the
        value 'realPort'.  If this PAE is not a Real Port, this object
        should be read only and returns 'false'.

        This object will be read only and returns 'false' if the value
```

```
        of the object ieee8021XPaePortCapabilities in the same row has
        the bit 'virtualPortsImplemented' off."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XPaePortEntry 8 }

ieee8021XPaePortMaxVirtualPorts OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The maximum number of virtual ports can be supported in this
        port."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 9 }

ieee8021XPaePortCurrentVirtualPorts OBJECT-TYPE
    SYNTAX          Gauge32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The current number of virtual ports is running in this port."
    REFERENCE       "802.1X Clause 12.9.2, Figure 12-3"
    ::= { ieee8021XPaePortEntry 10 }

ieee8021XPaePortVirtualPortStart OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if the virtual port is created by
        receipt of an EAPOL-Start packet."
    REFERENCE       "802.1X Clause 12.7, Figure 12-3"
    ::= { ieee8021XPaePortEntry 11 }

ieee8021XPaePortVirtualPortPeerMAC OBJECT-TYPE
    SYNTAX          MacAddress
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The source MAC address of the received EAPOL-Start if
        ieee8021XPaePortVirtualPortStart is set 'true'.

        If ieee8021XPaePortVirtualPortStart is not 'true' in the same
        row, the value of this object should be 00-00-00-00-00-00."
    REFERENCE       "802.1X Clause 12.7, Figure 12-3"
    ::= { ieee8021XPaePortEntry 12 }

ieee8021XPaePortLogonEnable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "Enable or disable to transmit network announcement
        information."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XPaePortEntry 13 }

ieee8021XPaePortAuthenticatorEnable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "True if the Authenticator is enabled.

        This object is read only. It returns 'false' if the value
        of the object ieee8021XPaePortCapabilities in the same row has
        the bit 'authImplemented' Off, or if the local control variable
        'enable' has not been set by the Logon Process."
    REFERENCE       "802.1X Clause 8.4 'enabled', Figure 12-3"
    ::= { ieee8021XPaePortEntry 14 }

ieee8021XPaePortSupplicantEnable OBJECT-TYPE
```

```
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "True if the Suppplicant is enabled.

        This object is read only. It returns 'false' if the PAE lacks
        supplicant functionality (ieee8021XPaePortCapabilities in the
        same row has the bit 'suppImplemented' off), or if the local
        control variable'enable' has not been set by the Logon Process
        (perhaps because the supplicant is designed to authenticate a
        human user and that user is not present)."
    REFERENCE       "802.1X Clause 8.4 'enabled', Figure 12-3"
    ::= { ieee8021XPaePortEntry 15 }

ieee8021XPaePortKayMkaEnable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "Enable or disable the MKA protocol function in this PAE.

        This object will be read only and returns 'false' if the value
        of the object ieee8021XPaePortCapabilities in the same row has
        the bit 'mkaImplemented' off."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XPaePortEntry 16 }

ieee8021XPaePortAnnouncerEnable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "Enable or disable the network Announcer function in this PAE.

        This object will be read only and returns 'false' if the value
        of the object ieee8021XPaePortCapabilities in the same row has
        the bit 'announcementsImplemented' off."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XPaePortEntry 17 }

ieee8021XPaePortListenerEnable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "Enable or disable the network Listener function in this PAE.

        This object will be read only and returns 'false' if the value
        of the object ieee8021XPaePortCapabilities in the same row has
        the bit 'listenerImplemented' off."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XPaePortEntry 18 }

ieee8021XPaeEapolGroupMAC OBJECT-TYPE
    SYNTAX          MacAddress
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The destination Group MAC Address used by this PAE when
        transmitting EAPOL frames."
    REFERENCE       "802.1X Clause 12.9, Figure 12-3"
    ::= { ieee8021XPaePortEntry 19 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAC Port Table
-- ---------------------------------------------------------------- --

ieee8021XPacPortTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XPacPortEntry
    MAX-ACCESS      not-accessible
```

```
    STATUS          current
    DESCRIPTION
        "A table of system level information for each interface
        supported by PAC.

        This table will be instantiated if the value of the object
        ieee8021XPaePortCapabilities in the corresponding entry of the
        ieee8021XPaePortTable has the bit 'macsecImplemented' off.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "IEEE 802.1X Clause 6.4, Clause 14"
    ::= { ieee8021XPaeSystem 6 }

ieee8021XPacPortEntry OBJECT-TYPE
    SYNTAX          Ieee8021XPacPortEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry containing PAC management information applicable to
        a particular interface."
    INDEX           { ieee8021XPacPortControlledPortNumber }
    ::= { ieee8021XPacPortTable 1 }

Ieee8021XPacPortEntry ::= SEQUENCE {
        ieee8021XPacPortControlledPortNumber    InterfaceIndex,
        ieee8021XPacPortAdminPt2PtMAC           INTEGER,
        ieee8021XPacPortOperPt2PtMAC            TruthValue
}

ieee8021XPacPortControlledPortNumber OBJECT-TYPE
    SYNTAX          InterfaceIndex
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The index to identify the 'Controlled Port' interface for a PAC."
    REFERENCE       "IEEE 802.1X Clause 6.4"
    ::= { ieee8021XPacPortEntry 1 }

ieee8021XPacPortAdminPt2PtMAC OBJECT-TYPE
    SYNTAX          INTEGER  {
                        forceTrue(1),
                        forceFalse(2),
                        auto(3)
                    }
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "An object to control the service connectivity to at most one
        other system.  The ieee8021XPacPortOperPt2PtMAC indicates
        operational status of the service connectivity for this PAC.

        'forceTrue' : allows only one service connection to the
                      other system.

        'forceFalse' : no restriction on the number of service
                       connections to the other systems.

        'auto' : means the service connectivity is determined by the
                 service providing entity."
    REFERENCE       "IEEE 802.1X Clause 6.4"
    DEFVAL          { auto }
    ::= { ieee8021XPacPortEntry 2 }

ieee8021XPacPortOperPt2PtMAC OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "An object to reflect the current service connectivity status.
```

```
        'true' : means the service connectivity of this PAC
             Controlled Port provides at most one other system.

        'false' : means the service connectivity of this PAC could
             provide more than one other system."
    REFERENCE       "IEEE 802.1X Clause 6.4"
    ::= { ieee8021XPacPortEntry 3 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAE Logon Process Group
-- ---------------------------------------------------------------- --
--
-- ---------------------------------------------------------------- --
-- The 802.1X PAE Logon Process Table
-- ---------------------------------------------------------------- --

ieee8021XPaePortLogonTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XPaePortLogonEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table of system level information for each port to support
        the Logon Process(es) status information.

        This table will be instantiated if the object
        ieee8021XPaePortLogonEnable in the corresponding entry of the
        ieee8021XPaePortTable is 'true'."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XPaeLogon 1 }

ieee8021XPaePortLogonEntry OBJECT-TYPE
    SYNTAX          Ieee8021XPaePortLogonEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains Logon Process status information for the
        PAE."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XPaePortLogonTable 1 }

Ieee8021XPaePortLogonEntry ::= SEQUENCE {
        ieee8021XPaePortLogonConnectStatus INTEGER,
        ieee8021XPaePortPortValid          TruthValue
}

ieee8021XPaePortLogonConnectStatus OBJECT-TYPE
    SYNTAX          INTEGER  {
                        pending(1),
                        unauthenticated(2),
                        authenticated(3),
                        secure(4)
                    }
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The Logon Process sets this variable to one of the following
        values, to indicate to the CP state machine if, and how,
        connectivity is to be provided through the Controlled Port :

        'pending' : Prevent connectivity by disabling the
            Controlled Port of this PAE.

        'unauthenticated' : Provide unsecured connectivity, enabling
            the Controlled Port of this PAE.

        'authenticated' : Provide unsecured connectivity but with
            authentication, enabling Controlled Port of this PAE.

        'secure' : Provide secure connectivity, using SAKs provided by
            the KaY (when available) and enabling Controlled Port when
```

```
            those keys are installed and in use."
    REFERENCE       "802.1X Clause 12.3, Figure 12-3"
    ::= { ieee8021XPaePortLogonEntry 1 }

ieee8021XPaePortPortValid OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' if Controlled Port communication
        is secured as specified by the MACsec."
    REFERENCE       "802.1X Clause 12.3, Figure 12-3"
    ::= { ieee8021XPaePortLogonEntry 2 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAE Session Table
-- ---------------------------------------------------------------- --

ieee8021XPaePortSessionTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XPaePortSessionEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table of system level information for each port to support
        Logon Process(es) session information.  This table maintains
        session statistics for its associated Controlled Port,
        suitable for communication to a RADIUS or other AAA server at
        the end of a session for accounting purpose.

        This table will be instantiated if the object
        ieee8021XPaePortLogonEnable in the corresponding entry of the
        ieee8021XPaePortTable is 'true'."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaeLogon 2 }

ieee8021XPaePortSessionEntry OBJECT-TYPE
    SYNTAX          Ieee8021XPaePortSessionEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains Logon Process session information for the
        PAE.  A session, an entry, begins when the operation of
        Controlled Port becomes 'true' and ends when it becomes
        'false'.

        The counts of frames and octets can be derived from those
        maintained to support from Interface MIB counters for the
        SecY's or the PAC's Controlled Port, but differs in that the
        counts are zeroed when the session begins."
    INDEX           { ieee8021XPaeSessionControlledPortNumber }
    ::= { ieee8021XPaePortSessionTable 1 }

Ieee8021XPaePortSessionEntry ::= SEQUENCE {
        ieee8021XPaeSessionControlledPortNumber   InterfaceIndex,
        ieee8021XPaePortSessionOctetsRx           Counter64,
        ieee8021XPaePortSessionOctetsTx           Counter64,
        ieee8021XPaePortSessionPktsRx             Counter64,
        ieee8021XPaePortSessionPktsTx             Counter64,
        ieee8021XPaePortSessionId                 SnmpAdminString,
        ieee8021XPaePortSessionStartTime          TimeStamp,
        ieee8021XPaePortSessionIntervalTime       TimeInterval,
        ieee8021XPaePortSessionTerminate          INTEGER,
        ieee8021XPaePortSessionUserName           SnmpAdminString
}

ieee8021XPaeSessionControlledPortNumber OBJECT-TYPE
    SYNTAX          InterfaceIndex
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The index to identify the 'Controlled Port' interface's session
```

```
        information for a PAE."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 1 }

ieee8021XPaePortSessionOctetsRx OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "Octets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of octets received in this session of this PAE.

        Discontinuities in the value of this counter can occur at
        re-initialization of the management system, and at
        other times as indicated by the value of
        ieee8021XPaePortSessionStartTime."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 2 }

ieee8021XPaePortSessionOctetsTx OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "Octets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of octets transmitted in this session of this PAE.

        Discontinuities in the value of this counter can occur at
        re-initialization of the management system, and at
        other times as indicated by the value of
        ieee8021XPaePortSessionStartTime."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 3 }

ieee8021XPaePortSessionPktsRx OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of packets received in this session of this PAE.

        Discontinuities in the value of this counter can occur at
        re-initialization of the management system, and at
        other times as indicated by the value of
        ieee8021XPaePortSessionStartTime."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 4 }

ieee8021XPaePortSessionPktsTx OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of packets transmitted in this session of this PAE.

        Discontinuities in the value of this counter can occur at
        re-initialization of the management system, and at
        other times as indicated by the value of
        ieee8021XPaePortSessionStartTime."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 5 }

ieee8021XPaePortSessionId OBJECT-TYPE
    SYNTAX          SnmpAdminString (SIZE  (3..253))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The session identifier for this session of the PAE.  A UTF-8
        string, uniquely identifying the session within the context of
        the PAE's system."
```

```
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 6 }

ieee8021XPaePortSessionStartTime OBJECT-TYPE
    SYNTAX          TimeStamp
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The starting time of this session."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 7 }

ieee8021XPaePortSessionIntervalTime OBJECT-TYPE
    SYNTAX          TimeInterval
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The duration time of the session has been last."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 8 }

ieee8021XPaePortSessionTerminate OBJECT-TYPE
    SYNTAX          INTEGER  {
                        macOperFailed(1),
                        sysAccessDisableOrPortInit(2),
                        receiveEapolLogOff(3),
                        eapReauthFailure(4),
                        mkaFailure(5),
                        newSessionBegin(6),
                        notTerminateYet(7)
                    }
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The reason for the session termination, one of the following :

        'macOperFailed' : 'Common Port' for this PAE is not
            operational.

        'sysAccessDisableOrPortInit' : The ieee8021XPaeSysAccessControl
            object is set to 'false' or initialization process of this
            PAE is invoked.

        'receiveEapolLogOff' : The PAE has received EAPOL-Logoff
            frame.

        'eapReauthFailure' : EAP reauthentication has failed.

        'mkaFailure' : MKA failure or other MKA termination.

        'newSessionBegin' : New session beginning.

        'notTerminateYet' : Not Terminated Yet."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 9 }

ieee8021XPaePortSessionUserName OBJECT-TYPE
    SYNTAX          SnmpAdminString (SIZE  (0..253))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The session user name for this session in the PAE.  A UTF-8
        string, representing the identity of the peer Supplicant.

        If no such information, zero length string will return."
    REFERENCE       "802.1X Clause 12.5.1, Figure 12-3"
    ::= { ieee8021XPaePortSessionEntry 10 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAE Logon Process NID Table
-- ---------------------------------------------------------------- --
```

```
ieee8021XLogonNIDTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XLogonNIDEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The Logon Process may use Network Identities (NIDs) to manage
        its use of authentication credentials, cached CAKs, and
        announcements.  This table provides the NID information for
        Logon Process.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XPaeLogon 3 }

ieee8021XLogonNIDEntry OBJECT-TYPE
    SYNTAX          Ieee8021XLogonNIDEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry provides the NID information for a Logon Process."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XLogonNIDTable 1 }

Ieee8021XLogonNIDEntry ::= SEQUENCE {
        ieee8021XLogonNIDConnectedNID Ieee8021XPaeNID,
        ieee8021XLogonNIDRequestedNID Ieee8021XPaeNIDOrNull,
        ieee8021XLogonNIDSelectedNID  Ieee8021XPaeNIDOrNull
}

ieee8021XLogonNIDConnectedNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNID
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The NID associated with the current connectivity (possibly
        unauthenticated) provided by the operation of the CP state
        machine.

        This object can differ from both the ieee8021XLogonNIDSelectedNID and
        the ieee8021XLogonNIDRequestedNID objects in the same row if
        authenticated connectivity (either secure or unsecured) has
        already been established, and EAP authentication and MKA
        operation for both of the latter have not met the necessary
        conditions (as specified by the control variables unauthAllowed
        and unsecureAllowed)."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XLogonNIDEntry 1 }

ieee8021XLogonNIDRequestedNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDOrNull
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The NID marked as access requested in announcements, as
        determined from EAPOL-Start frames.  The default of this object
        is as the configured value of object ieee8021XLogonNIDSelectedNID.

        This object information provides context for the PAE's EAP
        Authenticator.  If no EAPOL-Start frame has been received since
        the PAE's 'Common Port' became operational, or the last
        EAPOL-Start frame received for the port did not contain a
        requested NID, the object will take on the value of the object
        ieee8021XLogonNIDSelectedNID in the same row."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XLogonNIDEntry 2 }

ieee8021XLogonNIDSelectedNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDOrNull
```

```
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The NID currently configured for use by an access 'Controlled
        Port' when transmitting EAPOL-Start frames.  The default of
        this object is empty string.

        This object may be either explicitly configured by management
        or determined by the PAE using NID selection algorithms.  If no
        authentication is in progress, and the current connectivity is
        terminated and then starts again, ieee8021XLogonNIDConnectedNID will
        take on the value of ieee8021XLogonNIDRequestedNID (though a PAE
        NID's election algorithm, if used, can subsequently select
        another NID)."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    DEFVAL          { "" }
    ::= { ieee8021XLogonNIDEntry 3 }


-- ---------------------------------------------------------------- --
-- The PAE Authenticator Group
-- ---------------------------------------------------------------- --
--
-- ---------------------------------------------------------------- --
-- The 802.1X PAE Authenticator Table
-- ---------------------------------------------------------------- --

ieee8021XAuthenticatorTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XAuthenticatorEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table that contains the configuration objects for the
        Authenticator PAE associated with each port.  This table will
        be instantiated if the object ieee8021XPaePortAuthenticatorEnable in
        the corresponding entry of the ieee8021XPaePortTable is 'true'.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "802.1X Clause 8, Figure 12-3"
    ::= { ieee8021XPaeAuthenticator 1 }

ieee8021XAuthenticatorEntry OBJECT-TYPE
    SYNTAX          Ieee8021XAuthenticatorEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry that contains the Authenticator configuration objects
        for the PAE."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XAuthenticatorTable 1 }

Ieee8021XAuthenticatorEntry ::= SEQUENCE {
        ieee8021XAuthPaeAuthenticate  TruthValue,
        ieee8021XAuthPaeAuthenticated TruthValue,
        ieee8021XAuthPaeFailed        TruthValue,
        ieee8021XAuthPaeReAuthEnabled TruthValue,
        ieee8021XAuthPaeQuietPeriod   Unsigned32,
        ieee8021XAuthPaeReauthPeriod  Unsigned32,
        ieee8021XAuthPaeRetryMax      Unsigned32,
        ieee8021XAuthPaeRetryCount    Gauge32
}

ieee8021XAuthPaeAuthenticate OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' by the PAE authenticator to
        request authentication, and if this object is 'true',
```

```
        reauthentication is allowed.

        This object will be 'false' while the PAE authenticator revokes
        authentication."
    REFERENCE       "IEEE 802.1X Clause 8, Figure 12-3"
    ::= { ieee8021XAuthenticatorEntry 1 }

ieee8021XAuthPaeAuthenticated OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' by PACP if the PAE authenticator
        currently authenticated, and 'false' if the authentication
        fails or is revoked."
    REFERENCE       "IEEE 802.1X Clause 8, Figure 12-3"
    ::= { ieee8021XAuthenticatorEntry 2 }

ieee8021XAuthPaeFailed OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' by PACP if the authentication
        has failed or has been terminated.  The cause could be a
        failure returned by EAP, either immediately or following a
        reauthentication, an excessive number of attempts to
        authenticate (either immediately or upon reauthentication), or
        the authenticator deasserting authenticate, the object
        authPaeAuthenticate in the same row is 'false'.  The PACP
        will set the object authPaeAuthenticated false as well as
        setting the object 'true'."
    REFERENCE       "IEEE 802.1X Clause 8, Figure 12-3"
    ::= { ieee8021XAuthenticatorEntry 3 }

ieee8021XAuthPaeReAuthEnabled OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "This object is set 'true' if PACP should initiate
        reauthentication periodically, 'false' otherwise."
    REFERENCE       "IEEE 802.1X Clause 8.9, Figure 12-3"
    ::= { ieee8021XAuthenticatorEntry 4 }

ieee8021XAuthPaeQuietPeriod OBJECT-TYPE
    SYNTAX          Unsigned32 (0..65535)
    UNITS           "seconds"
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "This object indicates a waiting period after a failed
        authentication attempt, before another attempt is permitted."
    REFERENCE       "IEEE 802.1X Clause 8.6, Figure 12-3"
    DEFVAL          { 60 }
    ::= { ieee8021XAuthenticatorEntry 5 }

ieee8021XAuthPaeReauthPeriod OBJECT-TYPE
    SYNTAX          Unsigned32 (0..65535)
    UNITS           "seconds"
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "This object indicates the time period of the reauthentication
        to the supplicant."
    REFERENCE       "IEEE 802.1X Clause 8.6, Figure 12-3"
    DEFVAL          { 3600 }
    ::= { ieee8021XAuthenticatorEntry 6 }

ieee8021XAuthPaeRetryMax OBJECT-TYPE
    SYNTAX          Unsigned32
    UNITS           "times"
```

```
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The maximum number of authentication attempts before failure is
        reported to the Logon Process, and the authPaeQuietPeriod
        timer imposed before further attempts are permitted."
    REFERENCE       "IEEE 802.1X Clause 8.9, Figure 12-3"
    DEFVAL          { 2 }
    ::= { ieee8021XAuthenticatorEntry 7 }

ieee8021XAuthPaeRetryCount OBJECT-TYPE
    SYNTAX          Gauge32
    UNITS           "times"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The count of the number of authentication attempts."
    REFERENCE       "IEEE 802.1X Clause 8.9"
    ::= { ieee8021XAuthenticatorEntry 8 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAE Supplicant Group
-- ---------------------------------------------------------------- --
--
-- ---------------------------------------------------------------- --
-- The 802.1X PAE Supplicant Table
-- ---------------------------------------------------------------- --

ieee8021XSupplicantTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XSupplicantEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table that contains the configuration objects for the
        Supplicant PAE associated with each port.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "802.1X Clause 8, Figure 8-6, Figure 12-3"
    ::= { ieee8021XPaeSupplicant 1 }

ieee8021XSupplicantEntry OBJECT-TYPE
    SYNTAX          Ieee8021XSupplicantEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The configuration information for an Supplicant PAE."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XSupplicantTable 1 }

Ieee8021XSupplicantEntry ::= SEQUENCE {
        ieee8021XSuppPaeAuthenticate  TruthValue,
        ieee8021XSuppPaeAuthenticated TruthValue,
        ieee8021XSuppPaeFailed        TruthValue,
        ieee8021XSuppPaeHelloPeriod   Unsigned32,
        ieee8021XSuppPaeRetryMax      Unsigned32,
        ieee8021XSuppPaeRetryCount    Gauge32
}

ieee8021XSuppPaeAuthenticate OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' by the PAE supplicant to request
        authentication, and if this object is 'true', reauthentication
        is allowed.

        This object will be 'false' while the PAE supplicant revokes
```

```
        authentication."
    REFERENCE       "IEEE 802.1X Clause 8.4, Figure 8-6, Figure 12-3"
    ::= { ieee8021XSupplicantEntry 1 }

ieee8021XSuppPaeAuthenticated OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' by PACP if the PAE supplicant
        currently authenticated, and 'false' if the authentication
        fails or is revoked."
    REFERENCE       "IEEE 802.1X Clause 8.4, Figure 8-6, Figure 12-3"
    ::= { ieee8021XSupplicantEntry 2 }

ieee8021XSuppPaeFailed OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' by PACP if the authentication
        has failed or has been terminated.  The cause could be a
        failure returned by EAP, either immediately or following a
        reauthentication, an excessive number of attempts to
        authenticate (either immediately or upon reauthentication), or
        the supplicant deasserting authenticate, the object
        ieee8021XSuppPaeAuthenticate in the same row is 'false'.  The PACP
        will set the object ieee8021XSuppPaeAuthenticated false as well as
        setting the object 'true'."
    REFERENCE       "IEEE 802.1X Clause 8.4, Figure 8-6, Figure 12-3"
    ::= { ieee8021XSupplicantEntry 3 }

ieee8021XSuppPaeHelloPeriod OBJECT-TYPE
    SYNTAX          Unsigned32 (0..65535)
    UNITS           "seconds"
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The HeldPeriod before a failed authentication attempt is retried.
        Name HelloPeriod in previous MIB revisions retained for compatibility."
    REFERENCE       "IEEE 802.1X Clause 8.6, Figure 8-6, Figure 12-3"
    DEFVAL          { 60 }
    ::= { ieee8021XSupplicantEntry 4 }

ieee8021XSuppPaeRetryMax OBJECT-TYPE
    SYNTAX          Unsigned32
    UNITS           "times"
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The maximum number of authentication attempts before failure is
        reported to the Logon Process, and the ieee8021XSuppPaeHelloPeriod
        (HeldPeriod) imposed before further attempts are permitted."
    REFERENCE       "IEEE 802.1X Clause 8.7, Figure 8-6, Figure 12-3"
    DEFVAL          { 2 }
    ::= { ieee8021XSupplicantEntry 5 }

ieee8021XSuppPaeRetryCount OBJECT-TYPE
    SYNTAX          Gauge32
    UNITS           "times"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The count of the number of authentication attempts."
    REFERENCE       "IEEE 802.1X Clause 8.7, Figure 8-6, Figure 12-3"
    ::= { ieee8021XSupplicantEntry 6 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAE EAPOL Statistics Table
-- ---------------------------------------------------------------- --
```

```
ieee8021XEapolStatsTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XEapolStatsEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table in system level contains the EAPOL statistics and
        diagnostics information supported by PAE."
    REFERENCE       "802.1X Clause 12.8, Figure 12-3"
    ::= { ieee8021XPaeEapol 1 }

ieee8021XEapolStatsEntry OBJECT-TYPE
    SYNTAX          Ieee8021XEapolStatsEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains the EAPOL statistics and diagnostics
        information for a PAE."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XEapolStatsTable 1 }

Ieee8021XEapolStatsEntry ::= SEQUENCE {
        ieee8021XEapolInvalidFramesRx         Counter32,
        ieee8021XEapolEapLengthErrorFramesRx  Counter32,
        ieee8021XEapolAnnouncementFramesRx    Counter32,
        ieee8021XEapolAnnouncementReqFramesRx Counter32,
        ieee8021XEapolPortUnavailableFramesRx Counter32,
        ieee8021XEapolStartFramesRx           Counter32,
        ieee8021XEapolEapFramesRx             Counter32,
        ieee8021XEapolLogoffFramesRx          Counter32,
        ieee8021XEapolMkNoCknFramesRx         Counter32,
        ieee8021XEapolMkInvalidFramesRx       Counter32,
        ieee8021XEapolLastRxFrameVersion      Unsigned32,
        ieee8021XEapolLastRxFrameSource       MacAddress,
        ieee8021XEapolSuppEapFramesTx         Counter32,
        ieee8021XEapolLogoffFramesTx          Counter32,
        ieee8021XEapolAnnouncementFramesTx    Counter32,
        ieee8021XEapolAnnouncementReqFramesTx Counter32,
        ieee8021XEapolStartFramesTx           Counter32,
        ieee8021XEapolAuthEapFramesTx         Counter32,
        ieee8021XEapolMkaFramesTx             Counter32
}

ieee8021XEapolInvalidFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of invalid EAPOL frames of any type that have been
        received by this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 1 }

ieee8021XEapolEapLengthErrorFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL frames that the Packet Body Length does not
        match a Packet Body that is contained within the octets of the
        received EAPOL MPDU in this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 2 }

ieee8021XEapolAnnouncementFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Announcement frames that have been received
```

```
        by this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 3 }

ieee8021XEapolAnnouncementReqFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Announcement-Req frames that have been
        received by this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 4 }

ieee8021XEapolPortUnavailableFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL frames that are discarded because their
        processing would require the creation of a virtual port, for
        which there are inadequate or constrained resources, or an
        existing virtual port and no such port currently exists.  If
        virtual port is not supported, this object should be always 0."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 5 }

ieee8021XEapolStartFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Start frames that have been received by
        this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 6 }

ieee8021XEapolEapFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-EAP frames that have been received by
        this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 7 }

ieee8021XEapolLogoffFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Logoff frames that have been received by
        this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 8 }

ieee8021XEapolMkNoCknFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of MKPDUs received with MKA not enabled or CKN not
        recognized in this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 9 }
```

```
ieee8021XEapolMkInvalidFramesRx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of MKPDUs failing in message authentication on
        receipt process in this PAE."
    REFERENCE       "802.1X Clause 12.8.1, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 10 }

ieee8021XEapolLastRxFrameVersion OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The version of last received EAPOL frame by this PAE."
    REFERENCE       "802.1X Clause 12.8.2, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 11 }

ieee8021XEapolLastRxFrameSource OBJECT-TYPE
    SYNTAX          MacAddress
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The source MAC address of last received EAPOL frame by this
        PAE."
    REFERENCE       "802.1X Clause 12.8.2, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 12 }

ieee8021XEapolSuppEapFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-EAP frames that have been transmitted by
        the supplicant of this PAE."
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 13 }

ieee8021XEapolLogoffFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Logoff frames that have been transmitted by
        this PAE."
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 14 }

ieee8021XEapolAnnouncementFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Announcement frames that have been
        transmitted by this PAE."
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 15 }

ieee8021XEapolAnnouncementReqFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Announcement-Req frames that have been
        transmitted by this PAE."
```

```
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 16 }

ieee8021XEapolStartFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-Start frames that have been transmitted by
        this PAE."
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 17 }

ieee8021XEapolAuthEapFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-EAP frames that have been transmitted by
        the authenticator of this PAE."
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 18 }

ieee8021XEapolMkaFramesTx OBJECT-TYPE
    SYNTAX          Counter32
    UNITS           "Packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of EAPOL-MKA frames with no CKN information that
        have been transmitted by this PAE."
    REFERENCE       "802.1X Clause 12.8.3, Figure 12-3"
    ::= { ieee8021XEapolStatsEntry 19 }


-- ---------------------------------------------------------------------- --
-- The 802.1X PAE KaY Group
-- ---------------------------------------------------------------------- --
--
-- ---------------------------------------------------------------------- --
-- The 802.1X PAE KaY Table
-- ---------------------------------------------------------------------- --

ieee8021XKayMkaTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XKayMkaEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table of system level information for each interface
        supported by the KaY (Key Agreement Entity).  This table will
        be instantiated if the object ieee8021XPaePortKayMkaEnable in
        the corresponding entry of the ieee8021XPaePortTable is 'true'.

        The following terms are used to identify roles within the MKA
        protocol or protocol scenarios and the MIB description :

        participant : An instance of MKA, transmitting and receiving
            frames protected by keys derived from a single CAK, and
            operating with positive intent, obeying the protocol.

        member: A participant that possesses the CAK that can be used
            to prove liveness and to obtain membership in the CA under
            discussion.

        actor: The participant under discussion, usually in the KaY
            being described.

        partners: Participants or members attached to the same LAN as
            the actor, excluding the actor.
```

```
        principal actor: The actor controlling the PAC or SecY
            associated with the KaY.

        Each participant selects the live participant advertising the
        highest priority as its key server provided that participant
        has not selected another as its key server or is unwilling to
        act as the key server.  If a key server cannot be selected SAKs
        are not distributed.  In the event of a tie for highest
        priority key server, the member with the highest priority SCI
        is chosen.  For consistency with other uses of the SCI's MAC
        Address component as a priority, numerically lower values of
        the key server priority and SCI are accorded the highest
        priority.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XPaeKaY 1 }

ieee8021XKayMkaEntry OBJECT-TYPE
    SYNTAX          Ieee8021XKayMkaEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry containing KaY MKA management information applicable
        to a particular interface."
    INDEX           { ieee8021XPaePortNumber }
    ::= { ieee8021XKayMkaTable 1 }

Ieee8021XKayMkaEntry ::= SEQUENCE {
        ieee8021XKayMkaActive
            TruthValue,
        ieee8021XKayMkaAuthenticated
            TruthValue,
        ieee8021XKayMkaSecured
            TruthValue,
        ieee8021XKayMkaFailed
            TruthValue,
        ieee8021XKayMkaActorSCI
            SecySCI,
        ieee8021XKayMkaActorsPriority
            Ieee8021XMkaKeyServerPriority,
        ieee8021XKayMkaKeyServerPriority
            Ieee8021XMkaKeyServerPriority,
        ieee8021XKayMkaKeyServerSCI
            SecySCI,
        ieee8021XKayAllowedJoinGroup
            TruthValue,
        ieee8021XKayAllowedFormGroup
            TruthValue,
        ieee8021XKayCreateNewGroup
            TruthValue,
        ieee8021XKayMacSecCapability
            INTEGER,
        ieee8021XKayMacSecDesired
            TruthValue,
        ieee8021XKayMacSecProtect
            TruthValue,
        ieee8021XKayMacSecReplayProtect
            TruthValue,
        ieee8021XKayMacSecValidate
            TruthValue,
        ieee8021XKayMacSecConfidentialityOffset
            Integer32,
        ieee8021XKayMkaTxKN
            Ieee8021XMkaKN,
        ieee8021XKayMkaTxAN
            RowPointer,
        ieee8021XKayMkaRxKN
            Ieee8021XMkaKN,
```

```
        ieee8021XKayMkaRxAN
            RowPointer,
        ieee8021XKayMkaSuspendFor
            INTEGER,
        ieee8021XKayMkaSuspendOnRequest
            TruthValue,
        ieee8021XKayMkaSuspendedWhile
            INTEGER
}

ieee8021XKayMkaActive OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if there is at least one MKA active
        actor, transmitting MKPDUs"
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 1 }

ieee8021XKayMkaAuthenticated OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if the principal actor,
        i.e. the actor controlling the PAC or SecY associated with
        the KaY, has determined that Controlled Port communication
        communication should proceed without MACsec."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 2 }

ieee8021XKayMkaSecured OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if the principal actor has
        determined that communication should use MACsec."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 3 }

ieee8021XKayMkaFailed OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if the object
        ieee8021XKayMkaSecured in
        the same row is 'false' and MKA Life Time has elapsed since an
        MKA participant was last created."
    REFERENCE       "IEEE 802.1X Clause 9.16, Table 9-3, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 4 }

ieee8021XKayMkaActorSCI OBJECT-TYPE
    SYNTAX          SecySCI
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The SCI assigned by the system to the port, applies to all the
        port's MKA actors."
    REFERENCE
        "IEEE 802.1X Clause 9.16, Figure 12-3
         IEEE 802.1AE Clause 7.1.2, 10.7.1"
    ::= { ieee8021XKayMkaEntry 5 }

ieee8021XKayMkaActorsPriority OBJECT-TYPE
    SYNTAX          Ieee8021XMkaKeyServerPriority
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The Key Server priority for all the port's MKA actors.  Each
```

```
        participant encodes a key server priority, an 8-bit integer, in
        each MKPDU."
    REFERENCE       "IEEE 802.1X Clause 9.16, Table 9-2, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 6 }

ieee8021XKayMkaKeyServerPriority OBJECT-TYPE
    SYNTAX          Ieee8021XMkaKeyServerPriority
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The priority of the elected Key Server through MKA in the CA."
    REFERENCE       "IEEE 802.1X Clause 9.16, Table 9-2, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 7 }

ieee8021XKayMkaKeyServerSCI OBJECT-TYPE
    SYNTAX          SecySCI
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The SCI for key server for the MKA principal actor.  The length
        of this object is 0 if there is no principal actor, or that
        actor has no live peers.  This object matches the
        ieee8021XKayMkaActorSCI object in the same row if the actor is
        the key server."
    REFERENCE
        "IEEE 802.1X Clause 9.16, Figure 12-3
         IEEE 802.1AE Clause 7.1.2, 10.7.1"
    ::= { ieee8021XKayMkaEntry 8 }

ieee8021XKayAllowedJoinGroup OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if the KaY will accept Group CAKs
        distributed by MKA protocol."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 9 }

ieee8021XKayAllowedFormGroup OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object will be 'true' if the KaY will attempt to use
        point-to-point CAKs to distribute a group CAK, if it is the
        Key Server for the MKA instances for all the point-to-point CAKs."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 10 }

ieee8021XKayCreateNewGroup OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "This object is set 'true' if a new Group CAK is to be
        distributed if the KaY is the Key Server for the MKA instances
        for all the point-to-point CAKs.  This object will be set 'false'
        by the KaY when distribution is complete."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 11 }

ieee8021XKayMacSecCapability OBJECT-TYPE
    SYNTAX          INTEGER {
                        noMACsec(0),
                        macSecCapability1(1),
                        macSecCapability2(2),
                        macSecCapability3(3)
                    }
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
```

```
        "This object indicates whether MACsec is implemented, and if so
        whether the implementation provides integrity protection only,
        integrity and integrity with confidentiality, or integrity and
        integrity with confidentiality with a selectable confidentiality offset
        of 0, 30, or 50 octets (see IEEE Std 802.1AE).

        'noMACsec' : the MACsec is not implemented.

        'macSecCapability1' :  capable in 'integrity protection without
             confidentiality'.

        'macSecCapability2' :  capable in 'integrity protection without
             confidentiality' and integrity protection and confidentiali
              with a confidentiality offset 0',.

        'macSecCapability3' :  capable in 'integrity protection without
             confidentiality' and integrity protection and confidentiali
              with a confidentiality offset 0, 30 or 50'."
    REFERENCE
        "IEEE 802.1X Clause 9.6.1, Clause 9.16, Figure 12-3, Table 11-6"
    ::= { ieee8021XKayMkaEntry 12 }

ieee8021XKayMacSecDesired OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "This object will be set 'true' if the MKA participants desire
        the use of MACsec to protect frames with this KaY."
    REFERENCE
        "IEEE 802.1X Clause 9.6.1, Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 13 }

ieee8021XKayMacSecProtect OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The status of the MACsec protection function for this KaY.

        'true' : then the status of the MACsec protection function will
            be as object secyIfProtectFramesEnable object configured
            in the IEEE8021-SECY-MIB.
        'false' : then the MACsec protection function is disabled by
            this KaY."
    REFERENCE
        "IEEE 802.1X Clause 9.6.1, Clause 9.16, Figure 12-2,
         Figure 12-3, IEEE 802.1AE IEEE8021-SECY-MIB"
    ::= { ieee8021XKayMkaEntry 14 }

ieee8021XKayMacSecReplayProtect OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The status of the MACsec replay protection function for this
        KaY.

        'true' : then the status of the MACsec replay protection
            function will be as secyIfReplayProtectEnable object
            configured in the IEEE8021-SECY-MIB.
        'false' : then the MACsec replay protection function is
            disabled by this KaY."
    REFERENCE
        "IEEE 802.1X Clause 9.6.1, Clause 9.16, Figure 12-2,
         Figure 12-3"
    ::= { ieee8021XKayMkaEntry 15 }

ieee8021XKayMacSecValidate OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
```

```
    DESCRIPTION
        "The status of the MACsec validation function for this KaY.

        'true' : then the status of the MACsec validation function
            will be as secyIfValidateFrames object configured in the
            IEEE8021-SECY-MIB.
        'false' : then the MACsec validation function is enabled but
            only for checking without filtering out invalid frames by
            the SecY."
    REFERENCE
        "IEEE 802.1X Clause 9.6.1, Clause 9.16, Figure 12-2,
         Figure 12-3"
    ::= { ieee8021XKayMkaEntry 16 }

ieee8021XKayMacSecConfidentialityOffset OBJECT-TYPE
    SYNTAX          Integer32 (0 | 30 | 50)
    UNITS           "bytes"
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION
        "The confidentiality protection offset options for the selected
        cipher suite in the MACsec.  If the cipher suite does not have
        this capability, the configured value of the object will not
        apply to the cipher suite."
    REFERENCE
        "IEEE 802.1X Clause 9.7.1, Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 17 }

ieee8021XKayMkaTxKN OBJECT-TYPE
    SYNTAX          Ieee8021XMkaKN
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The key number assigned by the key server to the SAK currently
        being used for transmission.  This object will be 0 if MACsec
        is not being used or the key number is not available yet."
    REFERENCE       "IEEE 802.1X Clause 9.8, Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 18 }

ieee8021XKayMkaTxAN OBJECT-TYPE
    SYNTAX          RowPointer
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The AN assigned by the key server for use with the key number
        for transmission.

        This row pointer will point to an entry in the secyTxSATable
        which the secyTxSCEncodingSA object also points to in the
        IEEE8021-SECY-MIB.

        If MACsec is not in use or the AN is not identified yet, the
        value of this object shall be set to the OBJECT IDENTIFIER
        { 0 0 }."
    REFERENCE
        "IEEE 802.1X Clause 9.9, Clause 9.16, Figure 12-3,
         IEEE8021-SECY-MIB"
    ::= { ieee8021XKayMkaEntry 19 }

ieee8021XKayMkaRxKN OBJECT-TYPE
    SYNTAX          Ieee8021XMkaKN
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The key number assigned by the key server to the oldest SAK
        currently being used for reception.  It is the same as the key
        number for transmission if a single SAK is currently in use.
        This object will be 0 if MACsec is not being used or the key
        number is not available yet."
    REFERENCE       "IEEE 802.1X Clause 9.8, Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 20 }
```

```
ieee8021XKayMkaRxAN OBJECT-TYPE
    SYNTAX          RowPointer
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The AN assigned by the key server for use with the key number
        for reception.  It is the same as AN for transmission if a
        single SAK is currently in use.

        This row pointer will point to an entry in the secyRxSATable
        which the secyRxSCCurrentSA object also points to in the
        IEEE8021-SECY-MIB.

        If MACsec is not in use or the AN is not identified yet, the
        value of this object shall be set to the OBJECT IDENTIFIER
        { 0 0 }."
    REFERENCE
        "IEEE 802.1X Clause 9.6.1, Clause 9.16, Figure 12-3,
         IEEE8021-SECY-MIB"
    ::= { ieee8021XKayMkaEntry 21 }

ieee8021XKayMkaSuspendFor OBJECT-TYPE
    SYNTAX INTEGER (1..120)
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "Set by management to a non-zero number of seconds between 1
        and MKA Suspension Limit to initiate a suspension (9.18) of
        that duration (if the KaY's principal actor is the Key
        Server) or to request a suspension (otherwise)"
    REFERENCE "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 22 }

ieee8021XKayMkaSuspendOnRequest OBJECT-TYPE
    SYNTAX TruthValue
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "The status of the suspendOnRequest function for this KaY.
        'true' : then the KaY's principal actor will initiate a
        suspension if it is the Key Server and another participant
        has requested a suspension by transmitting a non-zero value
        of its suspendFor parameter
        'false' : then the KaY will not initiate a suspension on
        request from another participant."
    REFERENCE "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 23 }

ieee8021XKayMkaSuspendedWhile OBJECT-TYPE
    SYNTAX INTEGER (1..126)
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "Read by management to determine if a suspension is in
        progress and to discover the remaining duration of that
        suspension. May be set directly to coordinate in-service
        upgrades."
    REFERENCE "IEEE 802.1X Clause 5.11.4, Clause 9.16, Clause 9.18.5,
               Clause 9.18.6, Figure 12-3"
    ::= { ieee8021XKayMkaEntry 24 }

-- ------------------------------------------------------------------ --
-- The 802.1X PAE KaY MKA Participants Table
-- ------------------------------------------------------------------ --

ieee8021XKayMkaParticipantTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XKayMkaParticipantEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table for each MKA participant supported by the KaY MKA
        entity.
```

```
        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "IEEE 802.1X Clause 9.14, Clause 9.16, Figure 12-3"
    ::= { ieee8021XPaeKaY 2 }

ieee8021XKayMkaParticipantEntry OBJECT-TYPE
    SYNTAX          Ieee8021XKayMkaParticipantEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry containing KaY MKA management information applicable
        to a MKA participant."
    INDEX           { ieee8021XPaePortNumber, ieee8021XKayMkaPartCKN }
    ::= { ieee8021XKayMkaParticipantTable 1 }

Ieee8021XKayMkaParticipantEntry ::= SEQUENCE {
        ieee8021XKayMkaPartCKN             Ieee8021XPaeCKN,
        ieee8021XKayMkaPartKMD             Ieee8021XPaeKMD,
        ieee8021XKayMkaPartNID             Ieee8021XPaeNID,
        ieee8021XKayMkaPartCached          TruthValue,
        ieee8021XKayMkaPartActive          TruthValue,
        ieee8021XKayMkaPartRetain          TruthValue,
        ieee8021XKayMkaPartActivateControl INTEGER,
        ieee8021XKayMkaPartPrincipal       TruthValue,
        ieee8021XKayMkaPartDistCKN         Ieee8021XPaeCKNOrNull,
        ieee8021XKayMkaPartRowStatus       RowStatus
}

ieee8021XKayMkaPartCKN OBJECT-TYPE
    SYNTAX          Ieee8021XPaeCKN
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The CKN information for this MKA participant."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 1 }

ieee8021XKayMkaPartKMD OBJECT-TYPE
    SYNTAX          Ieee8021XPaeKMD
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The KMD information for this MKA participant."
    REFERENCE       "IEEE 802.1X Clause 9.16, Clause 12.6, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 2 }

ieee8021XKayMkaPartNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNID
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The NID information for this MKA participant."
    REFERENCE       "IEEE 802.1X Clause 9.16, Clause 12.6, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 3 }

ieee8021XKayMkaPartCached OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "This object is set 'true' by the KaY if the participant's
        parameters are cached.  If this object is 'true', this object
        can be set 'false' cleared by management to remove the
        participant's parameters from the cache."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 4 }

ieee8021XKayMkaPartActive OBJECT-TYPE
    SYNTAX          TruthValue
```

```
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object is set 'true' if the participant is active, i.e. is
        currently transmitting periodic MKPDUs."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    DEFVAL { false }
    ::= { ieee8021XKayMkaParticipantEntry 5 }

ieee8021XKayMkaPartRetain OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "This object is set 'true' to retain the participant in the
        cache, even if the KaY would normally remove it (due to lack
        of use for example)"
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 6 }

ieee8021XKayMkaPartActivateControl OBJECT-TYPE
    SYNTAX          INTEGER  {
                        default(1),
                        disabled(2),
                        onOperUp(3),
                        always(4)
                    }
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "This object is for controlling the participant's behavior when
        the participant is activated.

        'default' : the participant is from cached entries created by
            the KaY as part of normal operation, without explicit
            management, and is activated according to the
            implementation dependent policies of the KaY.

        'disabled' : the participant allows the cache information to
            be retained, but disabled for indefinite period.

        'onOperUp' : causing the participant to be activated when the
            PAE's 'Uncontrolled Port' becomes operational and when the
            PAE resumes following suspension.

        'always' : causing the participant to remain active all the
            time, even in the continued absence of partners.

        If the object changed to disabled(1) or onOperUp(3), the
        participant ceases operation immediately and receipt of MKPDUs
        with a matching CKN during a subsequent period of twice MKA
        lifetime will not cause the participant to become active once
        more."
    REFERENCE       "IEEE 802.1X Clause 9.14, Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 7 }

ieee8021XKayMkaPartPrincipal OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object is set 'true' if the participant is currently the
        principal actor."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    DEFVAL { false }
    ::= { ieee8021XKayMkaParticipantEntry 8 }

ieee8021XKayMkaPartDistCKN OBJECT-TYPE
    SYNTAX          Ieee8021XPaeCKNOrNull
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
```

```
        "The CKN for the last CAK distributed either by the actor or one
        of its partners.  Empty string for this object will be provided if
        this participant has not been used to distribute a CAK or the
        participant is not active, i.e. the object
        ieee8021XKayMkaPartActive in the same row is 'false'."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    DEFVAL { "" }
    ::= { ieee8021XKayMkaParticipantEntry 9 }

ieee8021XKayMkaPartRowStatus OBJECT-TYPE
    SYNTAX          RowStatus
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The object to create the parameters for the supported
        participant information in the system.

        If the participant information is from downloaded policies,
        this object is 'active'."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaParticipantEntry 10 }

-- ---------------------------------------------------------------- --
-- The 802.1X PAE MKA Peer List Table
-- ---------------------------------------------------------------- --

ieee8021XKayMkaPeerListTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XKayMkaPeerListEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table containing the lists of Live Peers and Potential Peers,
        for all MKA instances for which the KaY is active."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XPaeKaY 3 }

ieee8021XKayMkaPeerListEntry OBJECT-TYPE
    SYNTAX          Ieee8021XKayMkaPeerListEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table entry for one of the peers for one of the MKA
        instances for which this KaY is an active participant."
    INDEX           { ieee8021XPaePortNumber, ieee8021XKayMkaPartCKN,
                      ieee8021XKayMkaPeerListMI }
    ::= { ieee8021XKayMkaPeerListTable 1 }

Ieee8021XKayMkaPeerListEntry ::= SEQUENCE {
        ieee8021XKayMkaPeerListMI   Ieee8021XMkaMI,
        ieee8021XKayMkaPeerListMN   Ieee8021XMkaMN,
        ieee8021XKayMkaPeerListType INTEGER,
        ieee8021XKayMkaPeerListSCI  SecySCI
}

ieee8021XKayMkaPeerListMI OBJECT-TYPE
    SYNTAX          Ieee8021XMkaMI
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The peer entry's MI information in the peer list of this active
        participant in MKA protocol."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaPeerListEntry 1 }

ieee8021XKayMkaPeerListMN OBJECT-TYPE
    SYNTAX          Ieee8021XMkaMN
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The peer entry's latest MN information in the peer list of this
        active participant in MKA protocol."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
```

```
    ::= { ieee8021XKayMkaPeerListEntry 2 }

ieee8021XKayMkaPeerListType OBJECT-TYPE
    SYNTAX          INTEGER  {
                        livePeerList(1),
                        potentialPeerList(2)
                    }
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The peer entry's type in the peer list of this active
        participant in MKA protocol.

        'livePeerList' : the peer entry is in the Live Peer List.

        'potentialPeerList' : the peer entry is in the Potential
            Peer List."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaPeerListEntry 3 }

ieee8021XKayMkaPeerListSCI OBJECT-TYPE
    SYNTAX          SecySCI
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The SCI information of the peer entry in the peer list of this
        active participant in MKA protocol."
    REFERENCE       "IEEE 802.1X Clause 9.16, Figure 12-3"
    ::= { ieee8021XKayMkaPeerListEntry 4 }

-- ------------------------------------------------------------------ --
-- The 802.1X PAE NID Group
-- ------------------------------------------------------------------ --
--
-- ------------------------------------------------------------------ --
-- The 802.1X PAE NID Configuration Table
-- ------------------------------------------------------------------ --

ieee8021XNidConfigTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XNidConfigEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table that contains the configuration objects for the network
        announcement information for the Logon Process.

        The detail operation of the Logon Process can vary depending on
        the port-based network access control applications, and on the
        capabilities supported by that implementation including, for
        example, network discovery and roaming.  This table specifies
        control variables that facilitate behaviors that are
        potentially useful in a range of applications.  Implementations
        may use and augment the variables specified, or may use
        variables specific to the implementation.

        For the writeable objects in this table, the configured value
        shall be stored in persistent memory and remain unchanged
        across a re-initialization of the management system of the
        entity."
    REFERENCE       "802.1X Clause 8, Figure 8-6, Figure 12-3"
    ::= { ieee8021XPaeNetworkIdentifier 1 }

ieee8021XNidConfigEntry OBJECT-TYPE
    SYNTAX          Ieee8021XNidConfigEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains network announcement parameters for a NID."
    INDEX           { IMPLIED ieee8021XNidNID }
    ::= { ieee8021XNidConfigTable 1 }

Ieee8021XNidConfigEntry ::= SEQUENCE {
```

```
        ieee8021XNidNID                   Ieee8021XPaeNID,
        ieee8021XNidUseEap                INTEGER,
        ieee8021XNidUnauthAllowed         INTEGER,
        ieee8021XNidUnsecuredAllowed      INTEGER,
        ieee8021XNidUnauthenticatedAccess Ieee8021XPaeNIDUnauthenticatedStatus,
        ieee8021XNidAccessCapabilities    Ieee8021XPaeNIDCapabilites,
        ieee8021XNidKMD                   Ieee8021XPaeKMD,
        ieee8021XNidRowStatus             RowStatus
}

ieee8021XNidNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNID
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The network identifier to identify NID configuration in the
        PAE."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 1 }

ieee8021XNidUseEap OBJECT-TYPE
    SYNTAX          INTEGER  {
                        never(1),
                        immediate(2),
                        mkaFail(3)
                    }
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Determines when the Logon Process will initiate EAP, if the
        Supplicant and or Authenticator are enabled, and takes one of
        the following values:

        'never' : Never.

        'immediate' : Immediately, concurrently with the use of MKA
            with any cached CAK(s).

        'mkaFail' : Not until MKA has failed, if a prior CAK has been
            cached."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 2 }

ieee8021XNidUnauthAllowed OBJECT-TYPE
    SYNTAX          INTEGER  {
                        never(1),
                        immediate(2),
                        authFail(3)
                    }
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Determines when the Logon Process will tell the CP state
        machine to provide unauthenticated connectivity, and takes one
        of the following values:

        'never' : Never.

        'immediate' : Immediately, independently of any current or
            future attempts to authenticate using the PAE or MKA.

        'authFail' : Not until an attempt has been made to
            authenticate using EAP, unless neither the Supplicant nor
            the Authenticator is enabled, and MKA has attempted to use
            any cached CAK (unless the KaY is not enabled)."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 3 }

ieee8021XNidUnsecuredAllowed OBJECT-TYPE
    SYNTAX          INTEGER  {
                        never(1),
                        immediate(2),
```

```
                        mkaFail(3),
                        mkaServer(4)
                    }
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Determines when the Logon Process will tell the CP state
        machine to provide authenticated but unsecured connectivity,
        takes one of the following values:

        'never' : Never.

        'immediate' : Immediately, to provide connectivity
            concurrently with the use of MKA with any CAK acquired
            through EAP.

        'mkaFail' : Not until MKA has failed, or is not enabled.

        'mkaServer' : Only if directed by the MKA server."
    REFERENCE       "802.1X Clause 12.5, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 4 }

ieee8021XNidUnauthenticatedAccess OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDUnauthenticatedStatus
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The configured access capability of the port's clients without
        authentication in this NID."
    REFERENCE       "802.1X Clause 12.5, Clause 10.1, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 5 }

ieee8021XNidAccessCapabilities OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDCapabilites
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The authentication and protection capabilities supported for
        the NID."
    REFERENCE       "802.1X Clause 12.5, Clause 10.1, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 6 }

ieee8021XNidKMD OBJECT-TYPE
    SYNTAX          Ieee8021XPaeKMD
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The configured KMD information for this NID."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 7 }

ieee8021XNidRowStatus OBJECT-TYPE
    SYNTAX          RowStatus
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The object to create the parameters for the supported Network
        Announcement information in the system.

        If the Network Announcement information of the entry is from
        downloaded policies, this object is 'active'."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XNidConfigEntry 8 }


-- ---------------------------------------------------------- --
-- The 802.1X PAE Announce Information Table
-- ---------------------------------------------------------- --

ieee8021XAnnounceTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XAnnounceEntry
    MAX-ACCESS      not-accessible
```

```
    STATUS          current
    DESCRIPTION
        "A table contains the status information that the Announcers
        announce in the network announcement of the PAE system.

        This table will be instantiated if the object
        ieee8021XPaePortAnnouncerEnable in the corresponding entry of
        the ieee8021XPaePortTable is 'true'."
    REFERENCE       "802.1X Clause 8, Figure 8-6, Figure 12-3"
    ::= { ieee8021XPaeNetworkIdentifier 2 }

ieee8021XAnnounceEntry OBJECT-TYPE
    SYNTAX          Ieee8021XAnnounceEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains an Announcer's status information."
    INDEX           { ieee8021XPaePortNumber,
                      IMPLIED ieee8021XAnnounceNID }
    ::= { ieee8021XAnnounceTable 1 }

Ieee8021XAnnounceEntry ::= SEQUENCE {
        ieee8021XAnnounceNID          Ieee8021XPaeNID,
        ieee8021XAnnounceAccessStatus Ieee8021XPaeNIDAccessStatus
}

ieee8021XAnnounceNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNID
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The NID information to identify a transmitting network
        announcement for the PAE."
    REFERENCE       "802.1X Clause 10.4, Clause 12.5, Figure 12-3"
    ::= { ieee8021XAnnounceEntry 1 }

ieee8021XAnnounceAccessStatus OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDAccessStatus
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The object information reflects connectivity as a result of
        authentication attempts of this NID for this Announcer."
    REFERENCE
        "802.1X Clause 10.4, Clause 10.1, Clause 12.5, Figure 12-3"
    ::= { ieee8021XAnnounceEntry 2 }


-- ---------------------------------------------------------------- --
-- The 802.1X PAE Announcement Information Table
-- ---------------------------------------------------------------- --

ieee8021XAnnouncementTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XAnnouncementEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table contains the status information that the Listeners
        receive in the network announcement of the PAE system.

        This table will be instantiated if the object
        ieee8021XPaePortListenerEnable in the corresponding entry of the
        ieee8021XPaePortTable is 'true'."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XPaeNetworkIdentifier 3 }

ieee8021XAnnouncementEntry OBJECT-TYPE
    SYNTAX          Ieee8021XAnnouncementEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains a Listener's status information."
```

```
    INDEX           { ieee8021XPaePortNumber,
                      IMPLIED ieee8021XAnnouncementNID }
    ::= { ieee8021XAnnouncementTable 1 }

Ieee8021XAnnouncementEntry ::= SEQUENCE {
        ieee8021XAnnouncementNID             Ieee8021XPaeNID,
        ieee8021XAnnouncementKMD             Ieee8021XPaeKMD,
        ieee8021XAnnouncementSpecific        TruthValue,
        ieee8021XAnnouncementAccessStatus    Ieee8021XPaeNIDAccessStatus,
        ieee8021XAnnouncementAccessRequested TruthValue,
        ieee8021XAnnouncementUnauthAccess    Ieee8021XPaeNIDUnauthenticatedStatus,
        ieee8021XAnnouncementCapabilities    Ieee8021XPaeNIDCapabilites
}

ieee8021XAnnouncementNID OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNID
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The NID information to identify a received network announcement
        for the PAE."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 1 }

ieee8021XAnnouncementKMD OBJECT-TYPE
    SYNTAX          Ieee8021XPaeKMD
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The KMD information for this received network announcement of
        the PAE."
    REFERENCE       "802.1X Clause 10.4, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 2 }

ieee8021XAnnouncementSpecific OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object indicates the received announcement information was
        specific to the receiving PAE, not generic for all systems attached
        to the LAN."
    REFERENCE       "802.1X Clause 10.1, 10.4, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 3 }

ieee8021XAnnouncementAccessStatus OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDAccessStatus
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The object information reflects connectivity as a result of
        authentication attempts for this received network announcement
        of the PAE."
    REFERENCE       "802.1X Clause 10.4, Clause 10.1, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 4 }

ieee8021XAnnouncementAccessRequested OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The authenticated access has been requested for this particular
        NID or not."
    REFERENCE       "802.1X Clause 10.4, Clause 10.1, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 5 }

ieee8021XAnnouncementUnauthAccess OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDUnauthenticatedStatus
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The access capability of the port's clients without
```

```
        authentication in this received network announcement of the
        PAE.

        'openAccess', 'limitedAccess' should not be returned if the
        object ieee8021XNidUnauthAllowed is 'immediate'."
    REFERENCE
        "802.1X Clause 10.1, Clause 12.5, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 6 }

ieee8021XAnnouncementCapabilities OBJECT-TYPE
    SYNTAX          Ieee8021XPaeNIDCapabilites
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The announcement capabilities of this received network
        announcement for this PAE."
    REFERENCE       "802.1X Clause 10.1, Clause 12.5, Figure 12-3"
    ::= { ieee8021XAnnouncementEntry 7 }

-- ---------------------------------------------------------------- --
-- The 802.1X PAE Announcement Cipher Suite Information Table
-- ---------------------------------------------------------------- --

ieee8021XAnnouncementCipherSuitesTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF Ieee8021XAnnouncementCipherSuitesEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A table contains the Cipher Suites information that the Listeners
        receive in the network announcement of the PAE system.

        This table will be instantiated if the object
        ieee8021XPaePortListenerEnable in the corresponding entry of the
        ieee8021XPaePortTable is 'true'."
    REFERENCE       "802.1X Clause 10.4, Clause 11.13.3, Figure 11-21, Figure 12-3"
    ::= { ieee8021XPaeNetworkIdentifier 4 }

ieee8021XAnnouncementCipherSuitesEntry OBJECT-TYPE
    SYNTAX          Ieee8021XAnnouncementCipherSuitesEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry contains the Cipher Suite information which a Listener has
        reveived from network announcement."
    INDEX           { ieee8021XPaePortNumber,
                      ieee8021XAnnouncementNID,
                      ieee8021XAnnouncementCipherSuite }
    ::= { ieee8021XAnnouncementCipherSuitesTable 1 }

Ieee8021XAnnouncementCipherSuitesEntry ::= SEQUENCE {
        ieee8021XAnnouncementCipherSuite         OCTET STRING,
        ieee8021XAnnouncementCipherCapability    Unsigned32
}

ieee8021XAnnouncementCipherSuite OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE (8))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The identifier for the announced cipher suite.  This is a
        global unique 64-bit (EUI-64) identifier to identify a cipher
        suite."
    REFERENCE
        "802.1X Clause 10.4, Figure 12-3, 802.1AE-2006 Clause 14"
    ::= { ieee8021XAnnouncementCipherSuitesEntry 1 }

ieee8021XAnnouncementCipherCapability OBJECT-TYPE
    SYNTAX          Unsigned32 (0..65535)
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The capability of a Cipher Suite received from the network
```

```
        announcement by the Listener.

        A 2 octets Cipher Suite dependent implementation capability field
        precedes each Cipher Suite reference number. If the Cipher Suite,
        ieee8021XAnnouncementCipherSuite, identifies the Default Cipher
        Suite (specified in IEEE Std 802.1AE), the two least significant
        bits of the implementation capability field encode the MACsec
        Capability parameter specified in Table 11-7 and the fourteen more
        significant bits are as 0 and ignored on receipt."
    REFERENCE
        "802.1X Clause 11.13.3, Figure 11-21"
    ::= { ieee8021XAnnouncementCipherSuitesEntry 2 }

-- ---------------------------------------------------------------------- --
-- 802.1X Conformance
-- ---------------------------------------------------------------------- --

ieee8021XPaeCompliances  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBConformance 1 }

ieee8021XPaeGroups  OBJECT IDENTIFIER
    ::= { ieee8021XPaeMIBConformance 2 }


-- ---------------------------------------------------------------------- --
-- 802.1X Compliance Statements
-- ---------------------------------------------------------------------- --

ieee8021XPaeCompliance MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "The compliance statement for device support of
        Port Access Control."
    MODULE          -- this module
    MANDATORY-GROUPS {
                        ieee8021XPaeSystemGroup,
                        ieee8021XPaeLogonGroup,
                        ieee8021XPaeEapolStatsGroup
                    }

    GROUP           ieee8021XPacGroup
    DESCRIPTION
        "This group is mandatory for systems that do not support
        the MACsec functions of the PAE."

    GROUP           ieee8021XPaeAuthConfigGroup
    DESCRIPTION
        "This group is mandatory for systems that support the
        Authenticator functions of the PAE."

    GROUP           ieee8021XPaeSuppConfigGroup
    DESCRIPTION
        "This group is mandatory for systems that support the
        Supplicant functions of the PAE."

    GROUP           ieee8021XPaeKaYMkaGroup
    DESCRIPTION
        "This group is mandatory for systems that support the KaY
        MKA functions of the PAE."

    GROUP           ieee8021XPaeNetworkIdentifierGroup
    DESCRIPTION
        "This group is mandatory for systems that support the
        network announcement functions of the PAE."

    GROUP           ieee8021XPaeAnnouncerGroup
    DESCRIPTION
        "This group is mandatory for systems that support the
        network announcement and the Announcer functions of the
        PAE."

    GROUP           ieee8021XPaeListenerGroup
```

```
DESCRIPTION
    "This group is mandatory for systems that support
    the network announcement and the Listener functions of the
    PAE."

OBJECT          ieee8021XKayMacSecConfidentialityOffset
MIN-ACCESS      read-only
DESCRIPTION
    "read-write access is not required.  This may be read-only."

OBJECT          ieee8021XNidUseEap
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidUnauthAllowed
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidUnsecuredAllowed
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidUnauthenticatedAccess
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidAccessCapabilities
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidKMD
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidRowStatus
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."
::= { ieee8021XPaeCompliances 1 }

ieee8021XPaeV2Compliance MODULE-COMPLIANCE
STATUS          current
DESCRIPTION
    "The compliance statement for device support of
    Port Access Control as specified in 802.1X-2010
    amended by 802.1Xbx."
MODULE          -- this module
MANDATORY-GROUPS {
                    ieee8021XPaeSystemGroup,
                    ieee8021XPaeLogonGroup,
                    ieee8021XPaeEapolStatsGroup
                }

GROUP           ieee8021XPacGroup
DESCRIPTION
    "This group is mandatory for systems that does not support
    the MACsec functions of the PAE."

GROUP           ieee8021XPaeAuthConfigGroup
```

```
DESCRIPTION
    "This group is mandatory for systems that support the
    Authenticator functions of the PAE."

GROUP           ieee8021XPaeSuppConfigGroup
DESCRIPTION
    "This group is mandatory for systems that support the
    Supplicant functions of the PAE."

GROUP           ieee8021XPaeKaYMkaGroup
DESCRIPTION
    "This group is mandatory for systems that support the KaY
    MKA functions of the PAE."

GROUP           ieee8021XPaeNetworkIdentifierGroup
DESCRIPTION
    "This group is mandatory for systems that support the
    network announcement functions of the PAE."

GROUP           ieee8021XPaeAnnouncerGroup
DESCRIPTION
    "This group is mandatory for systems that support the
    network announcement and the Announcer functions of the
    PAE."

GROUP           ieee8021XPaeListenerGroup
DESCRIPTION
    "This group is mandatory for systems that support
    the network announcement and the Listener functions of the
    PAE."

GROUP           ieee8021XPaeKaYIsupgradeGroup
DESCRIPTION
    "This group is mandatory for systems that support KaY MKA
    in-service upgrades."

OBJECT          ieee8021XKayMacSecConfidentialityOffset
MIN-ACCESS      read-only
DESCRIPTION
    "read-write access is not required.  This may be read-only."

OBJECT          ieee8021XNidUseEap
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidUnauthAllowed
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidUnsecuredAllowed
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidUnauthenticatedAccess
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidAccessCapabilities
MIN-ACCESS      read-only
DESCRIPTION
    "read-create access is not required.  This may be
    read-only."

OBJECT          ieee8021XNidKMD
```

```
    MIN-ACCESS      read-only
    DESCRIPTION
        "read-create access is not required.  This may be
        read-only."

    OBJECT          ieee8021XNidRowStatus
    MIN-ACCESS      read-only
    DESCRIPTION
        "read-create access is not required.  This may be
        read-only."
    ::= { ieee8021XPaeCompliances 2 }


ieee8021XPaeSystemGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XPaeSysAccessControl,
                        ieee8021XPaeSysAnnouncements,
                        ieee8021XPaeSysEapolVersion,
                        ieee8021XPaeSysMkaVersion,
                        ieee8021XPaePortType,
                        ieee8021XPaeControlledPortNumber,
                        ieee8021XPaeUncontrolledPortNumber,
                        ieee8021XPaeCommonPortNumber,
                        ieee8021XPaePortInitialize,
                        ieee8021XPaePortCapabilities,
                        ieee8021XPaePortVirtualPortsEnable,
                        ieee8021XPaePortMaxVirtualPorts,
                        ieee8021XPaePortCurrentVirtualPorts,
                        ieee8021XPaePortVirtualPortStart,
                        ieee8021XPaePortVirtualPortPeerMAC,
                        ieee8021XPaePortLogonEnable,
                        ieee8021XPaePortAuthenticatorEnable,
                        ieee8021XPaePortSupplicantEnable,
                        ieee8021XPaePortKayMkaEnable,
                        ieee8021XPaePortAnnouncerEnable,
                        ieee8021XPaePortListenerEnable
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing system information for a PAE
        system and a PAE port status and control information."
    ::= { ieee8021XPaeGroups 1 }

ieee8021XPacGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XPacPortAdminPt2PtMAC,
                        ieee8021XPacPortOperPt2PtMAC
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing information of a PAC in the
        system."
    ::= { ieee8021XPaeGroups 2 }

ieee8021XPaeLogonGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XPaePortLogonConnectStatus,
                        ieee8021XPaePortPortValid,
                        ieee8021XPaePortSessionOctetsRx,
                        ieee8021XPaePortSessionOctetsTx,
                        ieee8021XPaePortSessionPktsRx,
                        ieee8021XPaePortSessionPktsTx,
                        ieee8021XPaePortSessionId,
                        ieee8021XPaePortSessionStartTime,
                        ieee8021XPaePortSessionIntervalTime,
                        ieee8021XPaePortSessionTerminate,
                        ieee8021XPaePortSessionUserName
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing information of a Logon
        Process in the system."
```

```
    ::= { ieee8021XPaeGroups 3 }

ieee8021XPaeAuthConfigGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XAuthPaeAuthenticate,
                        ieee8021XAuthPaeAuthenticated,
                        ieee8021XAuthPaeFailed,
                        ieee8021XAuthPaeReAuthEnabled,
                        ieee8021XAuthPaeQuietPeriod,
                        ieee8021XAuthPaeReauthPeriod,
                        ieee8021XAuthPaeRetryMax,
                        ieee8021XAuthPaeRetryCount
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing configuration information of
        an Authenticator in the system."
    ::= { ieee8021XPaeGroups 4 }

ieee8021XPaeSuppConfigGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XSuppPaeAuthenticate,
                        ieee8021XSuppPaeAuthenticated,
                        ieee8021XSuppPaeFailed,
                        ieee8021XSuppPaeHelloPeriod,
                        ieee8021XSuppPaeRetryMax,
                        ieee8021XSuppPaeRetryCount
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing configuration information of
        a Supplicant in the system."
    ::= { ieee8021XPaeGroups 5 }

ieee8021XPaeEapolStatsGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XEapolInvalidFramesRx,
                        ieee8021XEapolEapLengthErrorFramesRx,
                        ieee8021XEapolAnnouncementFramesRx,
                        ieee8021XEapolAnnouncementReqFramesRx,
                        ieee8021XEapolPortUnavailableFramesRx,
                        ieee8021XEapolStartFramesRx,
                        ieee8021XEapolEapFramesRx,
                        ieee8021XEapolLogoffFramesRx,
                        ieee8021XEapolMkNoCknFramesRx,
                        ieee8021XEapolMkInvalidFramesRx,
                        ieee8021XEapolLastRxFrameVersion,
                        ieee8021XEapolLastRxFrameSource,
                        ieee8021XEapolSuppEapFramesTx,
                        ieee8021XEapolLogoffFramesTx,
                        ieee8021XEapolAnnouncementFramesTx,
                        ieee8021XEapolAnnouncementReqFramesTx,
                        ieee8021XEapolStartFramesTx,
                        ieee8021XEapolAuthEapFramesTx,
                        ieee8021XEapolMkaFramesTx
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing counters and diagnostic
        information for the EAPOL in the system."
    ::= { ieee8021XPaeGroups 6 }

ieee8021XPaeKaYMkaGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XKayMkaActive,
                        ieee8021XKayMkaAuthenticated,
                        ieee8021XKayMkaSecured,
                        ieee8021XKayMkaFailed,
                        ieee8021XKayMkaActorSCI,
                        ieee8021XKayMkaActorsPriority,
                        ieee8021XKayMkaKeyServerPriority,
                        ieee8021XKayMkaKeyServerSCI,
```

```
                        ieee8021XKayAllowedJoinGroup,
                        ieee8021XKayAllowedFormGroup,
                        ieee8021XKayCreateNewGroup,
                        ieee8021XKayMacSecCapability,
                        ieee8021XKayMacSecDesired,
                        ieee8021XKayMacSecProtect,
                        ieee8021XKayMacSecReplayProtect,
                        ieee8021XKayMacSecValidate,
                        ieee8021XKayMacSecConfidentialityOffset,
                        ieee8021XKayMkaTxKN,
                        ieee8021XKayMkaTxAN,
                        ieee8021XKayMkaRxKN,
                        ieee8021XKayMkaRxAN,
                        ieee8021XKayMkaPartKMD,
                        ieee8021XKayMkaPartNID,
                        ieee8021XKayMkaPartCached,
                        ieee8021XKayMkaPartActive,
                        ieee8021XKayMkaPartRetain,
                        ieee8021XKayMkaPartActivateControl,
                        ieee8021XKayMkaPartPrincipal,
                        ieee8021XKayMkaPartDistCKN,
                        ieee8021XKayMkaPartRowStatus,
                        ieee8021XKayMkaPeerListMN,
                        ieee8021XKayMkaPeerListType,
                        ieee8021XKayMkaPeerListSCI
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing monitoring and controlling
        information of a KaY MKA in the system."
    ::= { ieee8021XPaeGroups 7 }

ieee8021XPaeNetworkIdentifierGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XLogonNIDConnectedNID,
                        ieee8021XLogonNIDRequestedNID,
                        ieee8021XLogonNIDSelectedNID,
                        ieee8021XNidUseEap,
                        ieee8021XNidUnauthAllowed,
                        ieee8021XNidUnsecuredAllowed,
                        ieee8021XNidUnauthenticatedAccess,
                        ieee8021XNidAccessCapabilities,
                        ieee8021XNidKMD,
                        ieee8021XNidRowStatus
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing monitoring and controlling
        information of an NID in the system."
    ::= { ieee8021XPaeGroups 8 }

ieee8021XPaeAnnouncerGroup OBJECT-GROUP
    OBJECTS         { ieee8021XAnnounceAccessStatus }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing status information for
        an Announcer in the system."
    ::= { ieee8021XPaeGroups 9 }

ieee8021XPaeListenerGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XAnnouncementKMD,
                        ieee8021XAnnouncementSpecific,
                        ieee8021XAnnouncementAccessStatus,
                        ieee8021XAnnouncementAccessRequested,
                        ieee8021XAnnouncementUnauthAccess,
                        ieee8021XAnnouncementCapabilities,
                        ieee8021XAnnouncementCipherCapability
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing status information for
```

```
        a Listener in the system."
    ::= { ieee8021XPaeGroups 10 }

ieee8021XPaeKaYIsupgradeGroup OBJECT-GROUP
    OBJECTS         {
                        ieee8021XKayMkaSuspendFor,
                        ieee8021XKayMkaSuspendOnRequest,
                        ieee8021XKayMkaSuspendedWhile
                    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing monitoring and control
        for MKA support of in-service upgrades."
    ::= { ieee8021XPaeGroups 11 }

ieee8021XPaeSystemAddGroup OBJECT-GROUP
       OBJECTS               {
                                          ieee8021XPaeEapolGroupMAC
                                      }
       STATUS                current
       DESCRIPTION
              "Objects previously overlooked, added by maintenance."
       ::= { ieee8021XPaeGroups 12 }

END
```

14. YANG Data Model

14.1 PAE management using YANG

This clause specifies a YANG data model that facilitates control and monitoring of the component protocol entities and processes for a system's PAEs by providing access to the operational controls, statistics, and diagnostic capabilities specified in 12.9 and summarized in the Unified Modeling Language (UML) information model in Figure 12-3. The data model also supports management of a system's PACs for access controlled ports that are not using MACsec.

NOTE 1—The MIBs specified in Clause 13 were also derived directly from 12.9 and Figure 12-3; therefore, the capabilities and structure of the YANG data model are closely aligned with that represented by the MIBs. However, the data model has not been derived from the MIB, and no attempt has been made to include data or modeling constructs that might appear in the MIB but not in the information model.

The development of Clause 14 has been guided by the YANG guidelines published in IETF RFC 6087 [B21] as applicable to IEEE standards.

Hierarchy has been introduced by the YANG framework in the following areas:

a) The uniform resource name (URN), as specified in IEEE Std 802d, has a structure where ieee is the root (i.e., name-space identifier), followed by the standard and then the working group developing the standard.

b) The YANG objects form a hierarchy of configuration and operational data structures that define the YANG model. These hierarchical relationships are described in 14.3.

c) The Interface YANG model is augmented by the PAE YANG sub-tree model. This provides PAE specific configuration and operational data extensions that can be associated with a designated interface.

Network interfaces are central to the management of protocols supported over the interface. Thus, it is important to establish a common data model for how interfaces are identified, configured, and monitored. The IETF Interface Management YANG data model (IETF RFC 8343) defines a generic YANG data model for the management of network interfaces. Additionally, common system level properties within a device (containing a network configuration protocol server) are provided in the IETF System Management YANG data model (IETF RFC 7317).

In general, interface type specific YANG data models (e.g., PAE) should augment the generic interfaces data model defined by the IETF Interface Management YANG data model (IETF RFC 8343). Additionally, system level properties within a device (such as a PAE System) should augment the generic system data model defined by the IETF System Management model (IETF RFC 7317).

NOTE 2—UML 2.5 [B34] conventions together with C++ language constructs are used to represent model structure and relationships.

It is expected that the PAE YANG modules and PAE MIBs will not co-exist as a means to manage PAEs and PAE Systems.

NOTE 3—This standard does not preclude the support of the PAE MIB and PAE YANG module at the same time on the device. However, it is expected that the user will use either YANG or the MIB for configuration and/or state retrieval. An implementation is not required to do both.

14.2 Security considerations

The YANG modules defined in this standard are designed to be accessed via the NETCONF protocol (IETF RFC 6241 [B22]). The lowest NETCONF layer is the secure transport layer. The lowest NETCONF layer is the secure transport and the mandatory to implement secure transport is SSH (IETF RFC 6242 [B23]). The NETCONF access control model (IETF RFC 6536 [B24]) provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

A number of data nodes defined in this YANG module are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. Listed below are the subtrees and data nodes and their sensitivity/vulnerability:

a) */system/pae-system/system-access-control*: Turning off network access control completely could render a network or a system accessing a network open to attack.

b) */interfaces/interface/pae/kay:enable*: Turning off MKA operation, would prevent use of MACsec and, in turn, either deny service or render communication open to attack.

c) */nid-group*: Modifying policies for unsecured or unauthenticated communication renders the system open to attack or denies the service (and thus encourages security to be disabled).

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. Listed below are the subtrees and data nodes and their sensitivity/vulnerability:

d) */system/pae-system/system-access-control*: Identify network access points that are not configured to secure access.

e) */nid-group*: Identify network access points that permit unauthenticated or unsecured access to certain network services.

MACsec could also be used to provide security for network configuration protocol functions when applied to this YANG model.

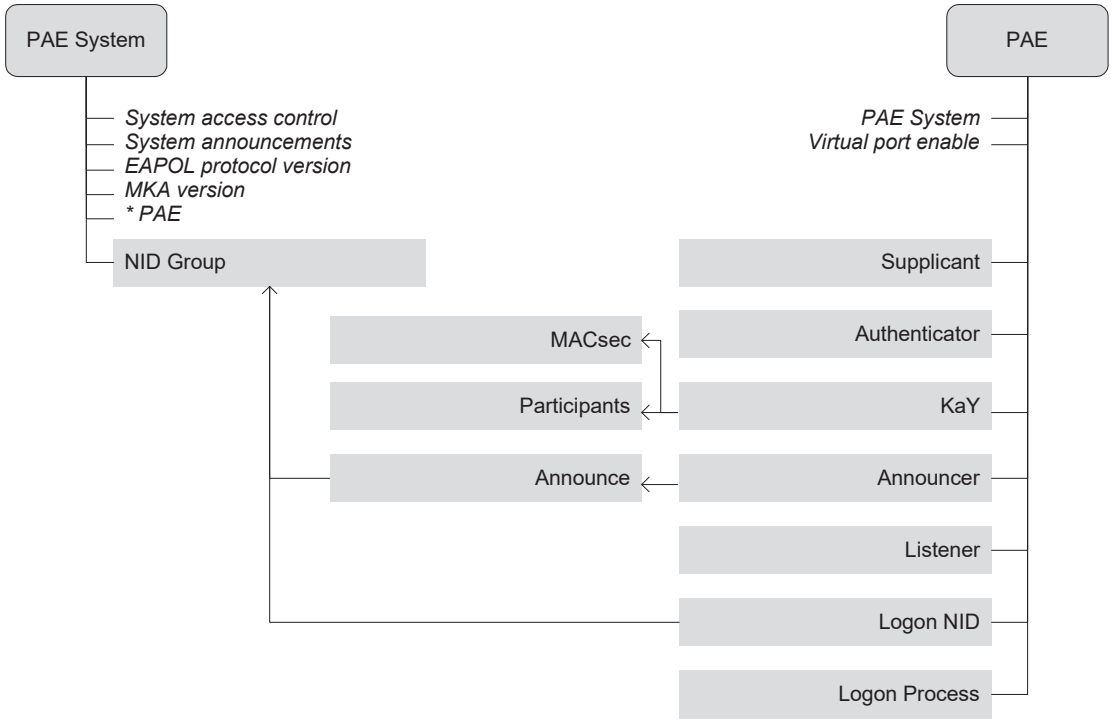
It is the responsibility of a system's implementor and administrator to ensure that the protocol entities in the system that support NETCONF, and any other remote configuration protocols that make use of these YANG modules, are properly configured to allow access only to the principals (users) that have legitimate rights to read or write data nodes. This standard does not specify how the credentials of those users are to be stored or validated.

The subject of this standard is network access control, and the YANG module(s) specified in Clause 14 that provide information can be presumed to include information that might be of use to an attacker, even if that attacker cannot directly control data that determines access rules or credentials. All the YANG module data nodes that are useful for operational management leak information about the configuration or performance of the system and might be used to direct attacks or evaluate their success. That being said, access to some data poses more significant risks and that data is the focus of this subclause (14.2).

This standard does not specify any management operations that provide read access to certain variables, such as secret keys, that are stored and used without being disclosed directly. Extensions to this data model, or to others in the system, that could provide such access are likely to compromise the security provided and could negate the purpose of this standard. Implementations of this standard should include mechanisms to ensure such keys can be used only for specified operations (such as calculating an ICV).

14.3 802.1X YANG model structure

A single YANG module is defined. The model comprises two primary YANG sub-trees: a PAE System sub-tree and a PAE sub-tree. The overall structure and the assignment of PAE objects to their groups are set out in Figure 14-1.



YANG nodes shaded in gray are containers that encompass additional leaf configuration or operational attributes. YANG nodes in white are leaf nodes that can be configurable (i.e., read-write) or operational (i.e., read-only) attributes.

Figure 14-1—YANG model structure

The PAE System YANG nodes augment the IETF System Management YANG data model (IETF RFC 7317), while the PAE YANG nodes augment the IETF Interface Management YANG data model (IETF RFC 8343), as illustrated in Figure 14-2.

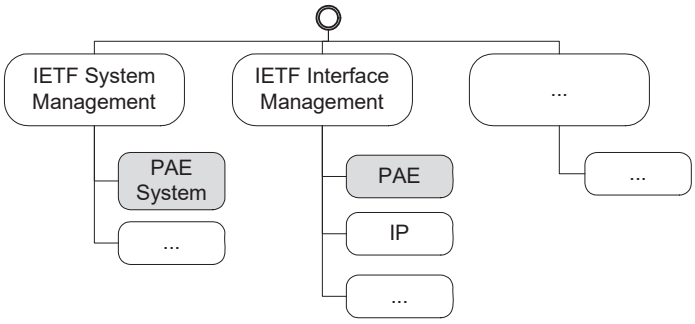


Figure 14-2—YANG object hierarchy with IEEE Std 802.1X

14.4 Relationship to other YANG data models

14.4.1 General

This standard specifies both a framework for port-based network control and particular details within that framework.

The IEEE 802.1X YANG data model augments the following key YANG models:

— IETF System Management YANG data model (IETF RFC 7317, 14.4.2)

— IETF Interface Management YANG data model (IETF RFC 8343, 14.4.3)

The use of the IEEE 802.1X YANG data model in a number of applications is described in 14.6. In some scenarios, that model is itself sufficient to meet the port-based network access control YANG data modeling requirements for particular devices. Stand-alone infrastructure devices can, for example, use pre-shared CAKs (PSKs) exclusively (i.e., require the use of MKA, but not EAP) and operate in scenarios where PSK installation by remote network management is not considered desirable. In other application scenarios, YANG data models can be required to support management of the following:

a) EAP, including the selection of EAP methods.

b) EAP credentials.

c) Trust roots to support EAP mutual authentication.

d) Provisioning and management of PSKs.

e) Policy based settings of other network parameters, depending on authorization associated with or following authentication, such as VLANs.

NOTE 1—At the time of publication of IEEE Std 802.1Xck-2018, there were no IEEE standard or IETF standards track RFC YANG data models addressing the requirements for a) through e) above.

Alternately, in some scenarios for some types of devices, one or more of these requirements might continue to be met by other management frameworks, as follows:

— For a personal device such as a laptop, some information [e.g., passwords used as EAP credentials, see b) above] might be entered directly at the keyboard by a human user, while other information can be displayed for checking. Equally, information supporting secure network access might be stored or captured for later reuse in the same way that the device handles other device settings. A device that is used for a range of tasks, e.g., both personal and work-related activities, might use information provided and controlled by a number of different management frameworks.

  NOTE 2—In the model of PAE operation (12.1, Figure 12-1), the Logon Process is responsible for controlling and coordinating the use of EAP, the CAK Cache, and MKA, and includes the acquisition and provision of the necessary credentials to EAP. The implementation of the Logon Process can vary to meet the needs of particular devices.

— Remote Authentication Dial-In User Service (RADIUS) (IETF RFC 2865 [B6], IETF RFC 3579 [B12]) can provide the EAP Authenticator with VLAN and other attribute settings (IETF RFC 3580 [B13], IETF RFC 4675 [B17]) to support e) above.

  NOTE 3—At the time of publication of IEEE Std 802.1Xck-2018, YANG models and YANG based management were not expected to supersede the use of RADIUS and RADIUS attributes, although a YANG model might facilitate the creation of policy profiles for use by EAP Authenticators and Supplicants or by CA members in scenarios where EAP is not used.

14.4.2 Relationship to the System Management YANG model

A system implementing the IEEE 802.1X YANG data model shall also implement the IETF System Management YANG data model defined in IETF RFC 7317. The IETF System Management YANG data model defines the configuration and identification of some common system properties within a device containing a network configuration protocol (e.g., NETCONF or other mechanisms). Figure 14-3 is a UML representation of the IETF System Management YANG data model.

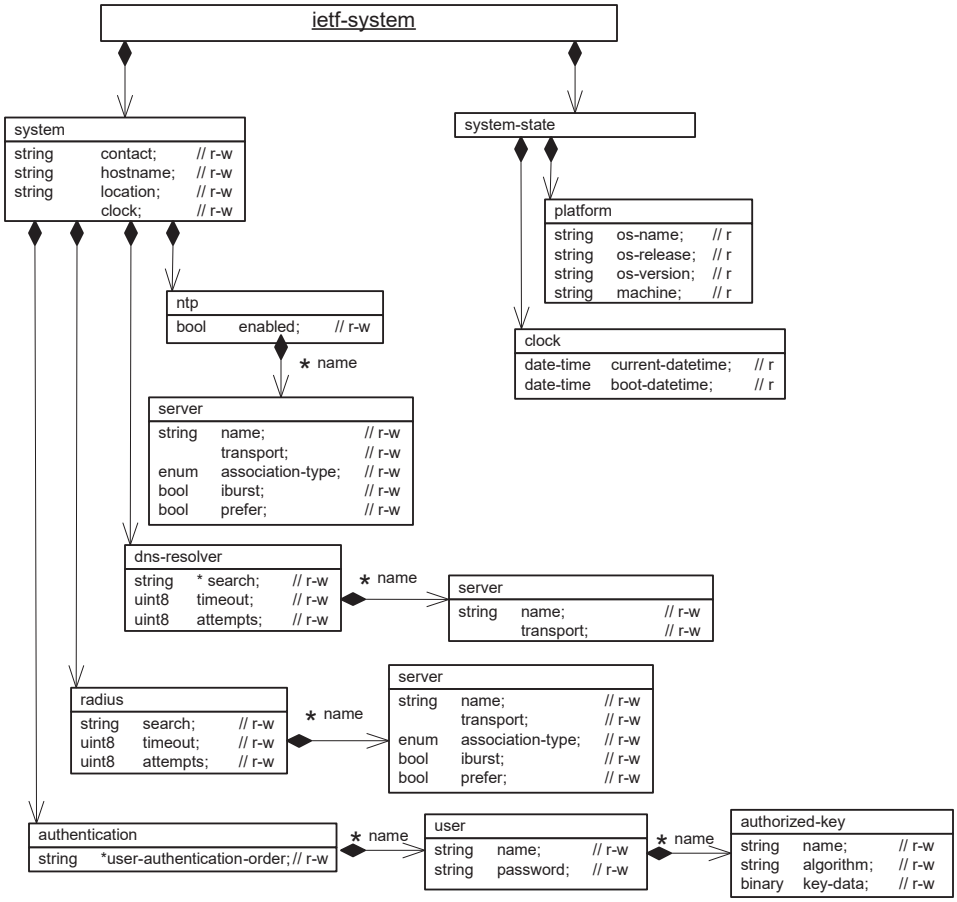


Figure 14-3—IETF System Management YANG data model

The PAE System YANG nodes augment the IETF System Management YANG data model with PAE system attributes. The system level PAE management entities outlined in 12.9.1 are contained within the PAE System YANG sub-tree. In addition, a list of PAEs that are governed by the PAE System is maintained, and a list of NIDs, which identifies the network services available to other systems attached to the same LAN, is included.

NOTE—A device can have multiple PAE Systems configured.

The PAE System cross-reference table in Table 14-1 associates the managed objects with the YANG module attributes.

Table 14-1—PAE System cross-reference table

PAE management information (Figure 12-3)	YANG node(s)
PAE System	**ietf-system:system:ieee802-dot1x:pae-system**
—	name
systemAccessControl (12.9.1) r-w	system-access-control
systemAnnouncements (12.9.1) r-w	system-announcements
eapolProtocolVersion (12.9.1, 11.3) r	eapol-protocol-version
mkaVersion (12.9.1, 11.3) r	mka-version
—	* pae[a]
*** NID**	**ietf-system:system:ieee802-dot1x:pae-system:nid-group**
nids (12.5)	nid — KEY
useEAP (12.5) r-w	use-eap
unauthAllowed (12.5) r-w	unauth-allowed
unsecureAllowed (12.5) r-w	unsecure-allowed
unauthenticated (12.5, 10.1) r-w	unauthenticated-access
accessCapabilities (12.5, 10.1) r-w	access-capabilities
kmd (10.4) r	kmd

[a] The asterisk (*) prior to an entity in this table represent a list of entities.

14.4.3 Relationship to the Interface Management YANG model

A system implementing the IEEE 802.1X YANG data model shall also implement the IETF Interface Management YANG data model defined in IETF RFC 8343. The IETF Interface Management YANG data model defines the management of network interfaces. Figure 14-4 is a UML representation of the IETF Interface Management YANG data model.

NOTE—Since network interfaces are central to the management of many Internet protocols, it is important to establish a common data model for how interfaces are identified, configured, and monitored.

The PAE YANG data model augments the IETF Interface Management YANG data model with PAE configuration and state data described in Figure 12-3. If the PAE utilizes a MAC Security Entity, then the interface stack would include a SecY shim. However, if there is no MAC Security Entity, then the interface stack would include a PAC (6.4), which is a protocol-less shim. The PAC managed objects specify attributes of a shim in an interface stack (IEEE Std 802.1AC). The Controlled Port and Uncontrolled Ports are service access points provided by the PAC. Following IETF RFC 2863, these two service access points are defined as supported by separate sublayers in the interface stack, and each has an individual conceptual Interface definition. The two interfaces are created together and co-exist without interference; one is not “on the top” of the other. The IETF Interface Management YANG data model leaf nodes *higher-layer-if* and *lower-layer-if* (as illustrated in Figure 14-4, interfaces-state object),[23] are used to identify the layer relationships between the (upper) Controlled Port interface provided by the PAC and the (lower) Common Port interface that it uses.

The YANG model can be used to manage an interface stack in one of two ways. The simplest management requirements can be met by augmenting an interface whose *ifType* (iana-if-type) is that of the PAC’s or

[23]For example, the interface stack relationship for a Link Aggregation Group (LAG) member, which is an Ethernet port, would be such that the *higher-layer-if* would reference the interface representing the LAG, while the *lower-layer-if* would reference the interface representing the Ethernet port.

SecY's Common Port (typically *ethernetCsmacd*). Frames transmitted and received by the PAE, modeled in this standard by its use of an Uncontrolled Port, are not subject to PAC or SecY control and do not contribute to the YANG Interface counters. Those interface counters reflect the use of the PAC or SecY's Controlled Port, and the *ifType* of the augmented interface is *macSecControlledIF*.[24,25]

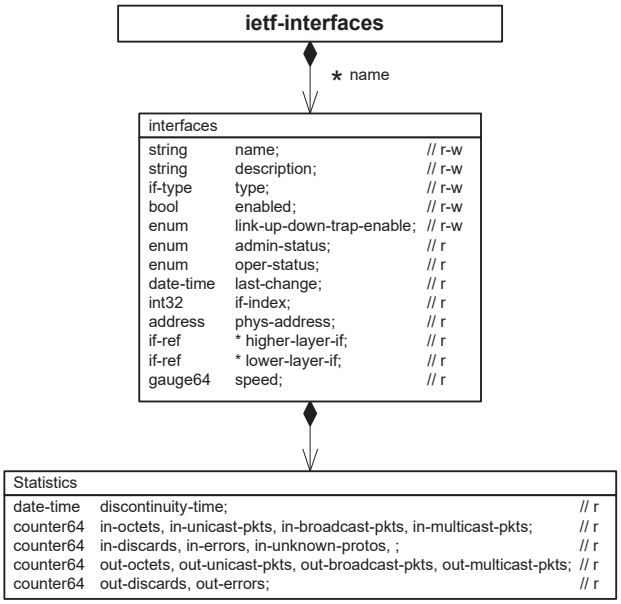


Figure 14-4—IETF Interface Management YANG data model

This simple model allows IEEE 802.1X functionality to be enabled or disabled on a port-by-port basis without any change to the interface stack and with the desirable attribute that the interface stack for an IEEE 802.1X capable Ethernet Port with IEEE 802.1X disabled (temporarily or permanently) does not differ from that for a port without IEEE 802.1X capability. Any further augmentation to the interface is assumed to be using the Controlled Port unless it is specified as using the Common Port or includes controls that specify which port it uses. In the absence of a suitable default or controls the following flexible management model needs to be used.

The MAC_Enabled and MAC_Operational parameters (6.4) are mapped to the *admin-status* and *oper-status* leaf operational nodes found within the Interface Management model.

Additionally, a pointer to the PAE System by which the PAE is governed is maintained. A PAE can be associated with an end station and/or MAC Bridges, where authentication, authorization, and secure communications (in accordance with this standard) are applied on one of its ports to any other system attached to the same LAN.

The PAE cross-reference table in Table 14-2 associates the managed objects with the YANG module attributes.

[24] Even if a PAC is used.

[25] When IEEE Std 802.1X is used in conjunction with IEEE Std 802.11, the interface type is as specified by IEEE 802.11 standards.

Table 14-2—PAE cross-reference table

PAE management information (Figure 12-3)	YANG node(s)
*** PAE[a]**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae**
—	pae-system
vpEnable (12.7) r-w	vp-enable
r	port-name
portNumber (12.9.2) r	port-number
r	controlled-port-name
controlledPortNumber (12.9.2) r	controlled-port-number
r	uncontrolled-port-name
uncontrolledPortNumber (12.9.2) r	uncontrolled-port-number
r	common-port-name
commonPortNumber (12.9.2) r	common-port-number
implemented.supp (12.9.2) r	port-capabilities:supp
implemented.auth (12.9.2) r	port-capabilities:auth
implemented.mka (12.9.2) r	port-capabilities:mka
implemented.macsec (12.9.2) r	port-capabilities:macsec
implemented.isupgrades (12.9.2) r	port-capabilities:in-service-upgrades
implemented.announcer (12.9.2) r	port-capabilities:annoucements
implemented.listener (12.9.2) r	port-capabilities:listener
implemented.virtualPorts (12.9.2) r	port-capabilities:virtual-ports
portType (12.9.2) r	port-type
maxVirtualPorts (12.9.2) r	virtual-port:max
currentVirtualPorts (12.9.2) r	virtual-port:current
vpStart (12.7) r	virtual-port:start
vpPeerAddress (12.7) r	virtual-port:peer-address
Supplicant	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:supplicant**
heldPeriod (8.6) r-w	held-period
retryMax (8.7) r-w	retry-max
enabled (8.4) r	enabled
authenticate (8.4) r	authenticated
authenticated (8.4) r	authenticated
failed (8.4) r	failed

Table 14-2—PAE cross-reference table (*continued*)

PAE management information (Figure 12-3)	YANG node(s)
Authenticator	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:authenticator**
quietPeriod (8.6) r-w	quiet-period
reauthPeriod (8.6) r-w	reauth-period
reauthEnable (8.1) r-w	reauth-enable
retryMax (8.9) r-w	retry-max
enabled (8.4) r	enabled
authenticate (8.4) r	authenticate
authenticated (8.4) r	authenticated
failed (8.4) r	failed
KaY	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:kay**
enable (9.16) r-w	enable
actorPriority (9.16) r-w	actor:priority
keyServerPriority (9.16) r-w	key-server:priority
joinGroup (9.16) r-w	group:join
formGroup (9.16) r-w	group:form
newGroup (9.16) r-w	group:new
macsecCapable (9.16) r-w	macsec:capable
macsecDesired (9.16) r-w	macsec:desired
suspendOnRequest (9.18) r-w	suspend-on-request
suspendFor (9.18) r-w	suspend-for
actorSCI (9.16) r	actor:sci
keyServerSCI (9.16) r	key-server:sci
macsecProtect (9.16) r	macsec:protect
macsecValidate (9.16) r	macsec:validate
macsecReplayProtect (9.16) r	macsec:replay-protect
suspendedWhile (9.18) r	suspended-while
active (9.16) r	active
authenticated (9.16) r	authenticated
secured (9.16) r	secured
failed (9.16) r	failed
txKN (9.16) r	key-number:tx
rxKN (9.16) r	key-number:rx
txAN (9.16) r	association-number:tx
rxAN (9.16) r	association-number:rx

Table 14-2—PAE cross-reference table (*continued*)

PAE management information (Figure 12-3)	YANG node(s)
*** Participant**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:kay:participant**
—	participants — KEY
cached (9.16) r-w	cached
active (9.16) r-w	active
retain (9.16) r-w	retain
activate (9.16) r-w	activate
livePeers (9.16) r	peers:*live
potentialPeers (9.16) r	peers:*potential
ckn (9.16) r	ckn
kmd (9.16) r	kmd
nid (9.16) r	nid
authData (9.16) r	auth-data
principal (9.16) r	principal
distCKN (9.16) r	dist-ckn
LogonNIDs	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:logon-nid**
selectedNID (12.5) r-w	selected
connectedNID (12.5) r	connected
requestedNID (12.5) r	requested
*** NID**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:logon-nid:nid-group**
connectedNID, selectedNID, requestedNID (12.5)	nid — KEY
useEAP (12.5) r-w	use-eap
unauthAllowed (12.5) r-w	unauth-allowed
unsecureAllowed (12.5) r-w	unsecure-allowed
unauthenticated (12.5, 10.1) r-w	unauthenticated-access
accessCapabilities (12.5, 10.1) r-w	access-capabilities
kmd (10.4) r	kmd

Table 14-2—PAE cross-reference table (*continued*)

PAE management information (Figure 12-3)	YANG node(s)
Announcer	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:announcer**
enable (10.4) r-w	enable
*** Announce**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:announcer.announce**
—	announces — KEY
nid (10.4) r	nid
accessStatus (10.4, 12.5) r	access-status
*** NID**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:announcer.announce:nid-group**
nid (12.5)	nid — KEY
useEAP (12.5) r-w	use-eap
unauthAllowed (12.5) r-w	unauth-allowed
unsecureAllowed (12.5) r-w	unsecure-allowed
unauthenticated (12.5, 10.1) r-w	unauthenticated-access
accessCapabilities (12.5, 10.1) r-w	access-capabilities
kmd (10.4) r	kmd
Listener	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:listener**
enable (10.4) r-w	enable
*** Announcement**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:listener:annoucement**
—	announcements — KEY
nid (10.4) r	nid
kmd (10.4) r	kmd
specific (10.4) r	specific
accessStatus (10.4) r	access-status
requestedNID (10.4) r	requested-nid
unauthenticatedAccess (10.4) r	unauthenticated-access
accessCapabilities (10.4) r	access-capabilities
* Ciphersuites (10.4) r	cipher-suites
—	index — KEY
—	cipherSuite
—	cipherSuiteCapability

Table 14-2—PAE cross-reference table (*continued*)

PAE management information (Figure 12-3)	YANG node(s)
EapolStatistics	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:eapol-statistics**
invalidEapolFramesRx (12.8.1) r	invalid-eapol-frame-rx
eapLengthErrorFrames (12.8.1) r	eap-length-error-frames
eapolAnnouncementsRx (12.8.1) r	eapol-announcements-rx
eapolAnnounceReqsRx (12.8.1) r	eapol-announce-reqs-rx
eapolPortUnavailable (12.8.1) r	eapol-port-unavailable
eapolStartFramesRx (12.8.1) r	eapol-start-frames-rx
eapolEapFramesRx (12.8.1) r	eapol-eap-frames-rx
eapolLogoffFramesRx (12.8.1) r	eapol-logoff-frames-rx
eapolMKnoCKN (12.8.1) r	eapol-mk-no-cfn
eapolMkInvalidFramesRx (12.8.1) r	eapol-mk-invalid-frames-rx
lastEapolFrameSource (12.8.2) r	last-eapol-frame-source
lastEapolFrameVersion (12.8.2) r	last-eapol-frame-version
eapolSuppEapFramesTx (12.8.3) r	eapol-supp-eap-frames-tx
eapolLogoffFramesTx (12.8.3) r	eapol-logoff-frames-tx
eapolAnnouncementsTx (12.8.3) r	eapol-announcements-tx
eapolAnnounceReqsTx (12.8.3) r	eapol-announce-reqs-tx
eapolStartFramesTx (12.8.3) r	eapol-start-frames-tx
eapolAuthEapFramesTx (12.8.3) r	eapol-auth-eap-frames-tx
eapolMKAFramesTx (12.8.3) r	eapol-mka-frames-tx
LogonProcess	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:logon-process**
logon (12.5) r-w	logon
connect (12.3) r	connect
portValid (12.3) r	port-valid

Table 14-2—PAE cross-reference table (*continued*)

PAE management information (Figure 12-3)	YANG node(s)
*** SessionStatistics**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:logon-process:session-statistics**
sessionId (12.5.1) r	session-id — KEY
sessionUserName (12.5.1) r	user-name
sessionOctetsRx (12.5.1) r	octets-rx
sessionOctetsTx (12.5.1) r	octets-tx
sessionFramesRx (12.5.1) r	frames-rx
sessionFramesTx (12.5.1) r	frames-tx
sessionTime (12.5.1) r	time
sessionTerminateCause (12.5.1) r	terminate-cause
*** NID**	**ietf-interfaces:interfaces:interface:ieee802-dot1x:pae:nid-group**
nids (12.5)	nid — KEY
useEAP (12.5) r-w	use-eap
unauthAllowed (12.5) r-w	unauth-allowed
unsecureAllowed (12.5) r-w	unsecure-allowed
unauthenticated (12.5, 10.1) r-w	unauthenticated-access
accessCapabilities (12.5, 10.1) r-w	access-capabilities
kmd (10.4) r	kmd

[a] The asterisk (*) prior to an entity in this table represent a list of entities.

14.4.4 The Interface Stack Models

The YANG model supports both explicit and augmented interface models. An explicitly modeled interface represents a single shim or sublayer in an interface stack. An augmented interface combines adjacent shims or sublayers into a single interface to simplify configuration. Figure 14-5, for example, represents the structure of a simple Bridge Port explicitly, using one interface to model the IEEE 802.3 attributes of the port and another to represent attributes associated with their use as a Bridge Port.The small alphanumeric circle overlaying the larger clear circles represent the index used to identify the Interface. Here, Interface A would be an ifType of *bridge*, with a lower-layer-if of Interface B, while Interface B would be an ifType of *ethernetCsmacd* with a higher-layer-if of Interface A.

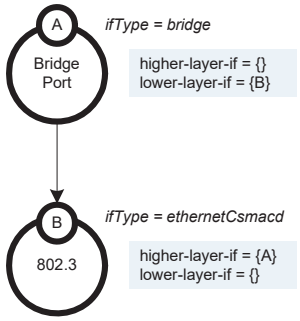


Figure 14-5—Explicit Interface Model of Bridge Port

Figure 14-6 shows the IEEE 802.3 interface, Interface A, augmented with Bridge Port configuration and operational data. The shaded circles present protocol entities, while the clear circles represent service access points (SAPs). Dashed ovals represent the set of protocol entities sharing the same Interface as the SAP.

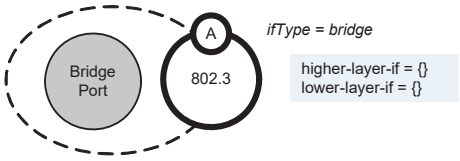


Figure 14-6—Augmented Interface Mode of Bridge Port.

NOTE—In Figure 14-6, it is assumed that the network manager has overwritten the ifType to reflect the service provided by the augmented interface, i.e., that of a Bridge Port.

The support of Link Aggregation (IEEE Std 802.1AX) will also introduce further variations in the Interface Stack that can be supported. For example, as illustrated in Figure 14-7, Interface A, which is a link aggregation port, would have an ifType of *bridge* because it is augmented by the Bridge Port. The link aggregation members are represented by Interface B and Interface C, both of which are of ifType *ethernetCsmacd*. The lower-layer-if of Interface A would comprised Interface B and Interface C. The higher-layer-if of Interface B and Interface C would be Interface A.

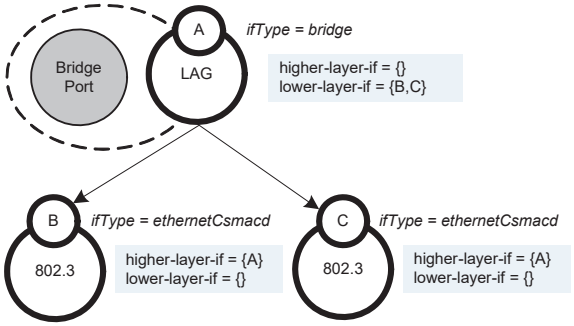


Figure 14-7—Bridge Port with LAG Interface stack model

More exacting management requirements are met with interface stacks that comprise two or more tiered YANG interfaces. For example, Figure 14-8 provides an illustration of the YANG model with MACsec. The service that provides the Common Port to a PAC or SecY is associated with an Interface and the Controlled Port; or, as in the case of Virtual Ports, the multiple Control Ports that make use of a single Common Port are separate YANG interfaces. This tiered interface stack is required for Virtual Port management and also allows use of the Controlled Port or the Common Port by other protocol entities (such as LLDP) to be specified without augmenting the YANG models for those entities.

The Interface with index of B, would have an ifType of *macSecControlledIF*. The higher-layer-if of Interface B would be Interface A, while the lower-layer-if is Interface C.

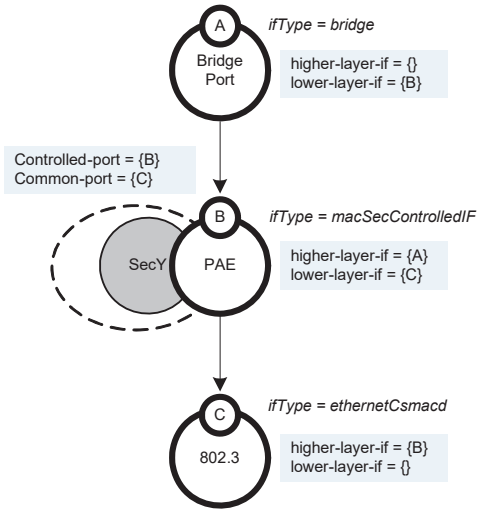


Figure 14-8—Bridge Port YANG Interface stack model with MACsec

A system can support one or both (e.g., explicit or augmented interface model) of these interface stack styles, but any change from one to another occurs only as a result of explicit management configuration, not as a side effect or as a result of dynamic conditions. The use and configuration of other YANG models can depend on the use of a simple, single-tiered, interface stack or a multi-tiered stack.

Figure 14-9 provides an illustration of an augmented interface of a Bridge Port with a PAE. The Bridge Port is the result of an augmentation of Interface A, which was of ifType *ethernetCsmacd*. Subsequently, the PAE augments Interface A.

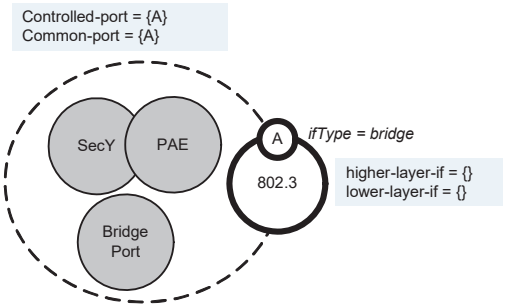


Figure 14-9—Augmented Interface Model of Bridge Port with MACsec

Figure 14-10 provides an example illustration of the YANG Interface model representing a real port and multiple virtual ports.

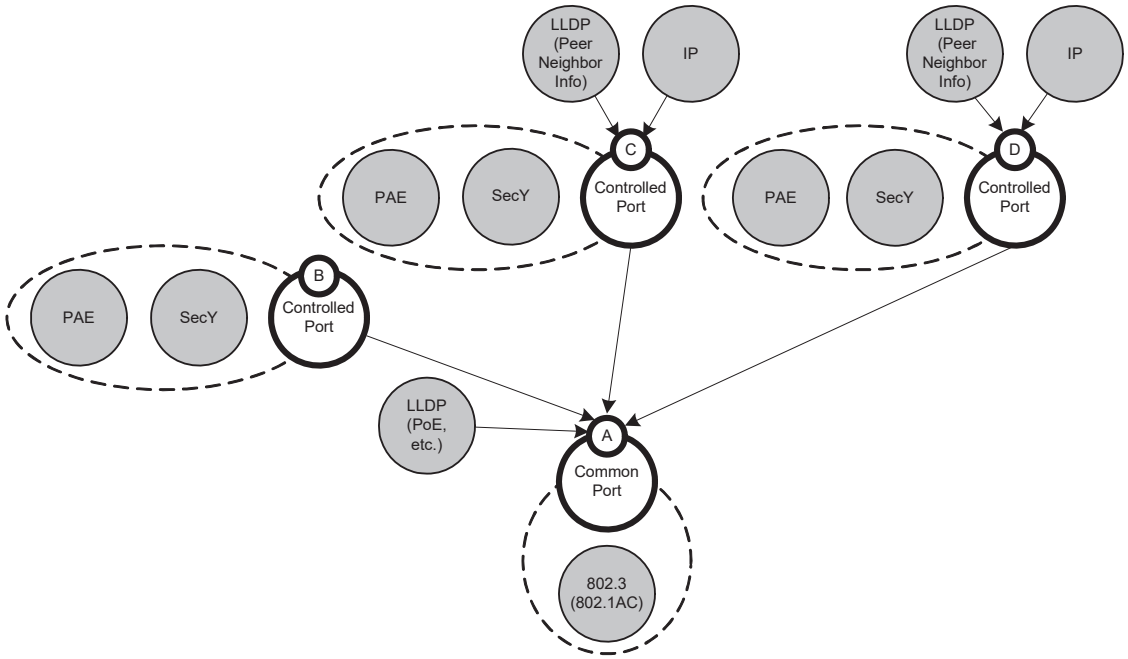


Figure 14-10—YANG Interface Model with MACsec and virtual ports

The introduction of MACsec on link aggregation members can be expressed as an explicit or augmented interface model. Figure 14-11 provides an illustration of interfaces being explicitly modeled within the interface stack of MACsec configured on aggregation members of a Bridge Port that is a LAG. Interface A of ifType *bridge* is a LAG interface augmented by Bridge Port. PAE Interface B and Interface M are inserted above Interface C and Interface N, respectively, in the stack. The Controlled-ports are Interface B and Interface M, each of which have an ifType of *macSecControlledIF*. The interface stack is very clear and explicit.

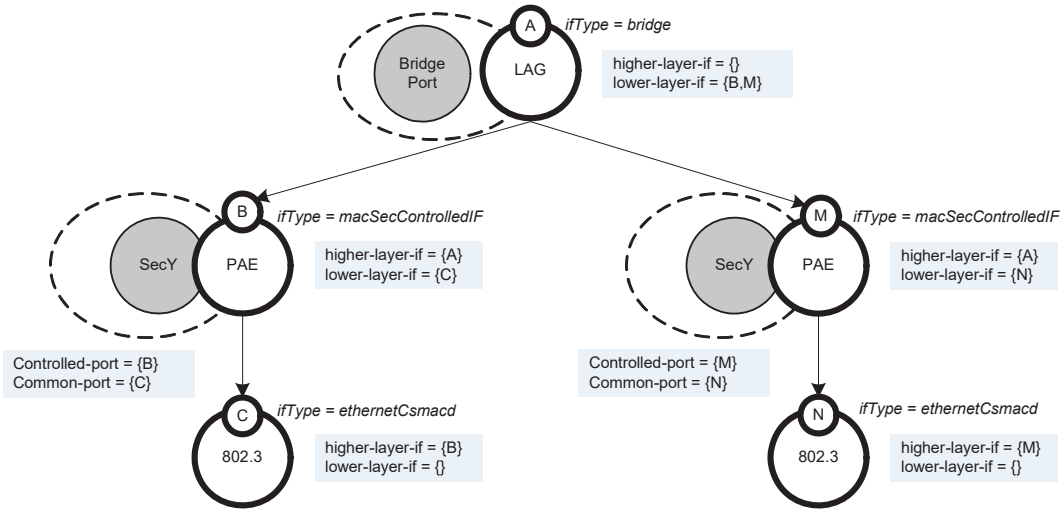


Figure 14-11—Explicit Interface Model of Bridge Port LAG with MACsec on members

Conversely, Figure 14-12 provides an illustration of an augmented interface model where the interface stack represents MACsec on aggregation members of a Bridge Port that is a link aggregation group. Interface A of ifType *bridge* is an LAG interface augmented by Bridge Port. The PAE augments Interface B and Interface C, respectively, each of which have an ifType of *ethernetCsmacd*.

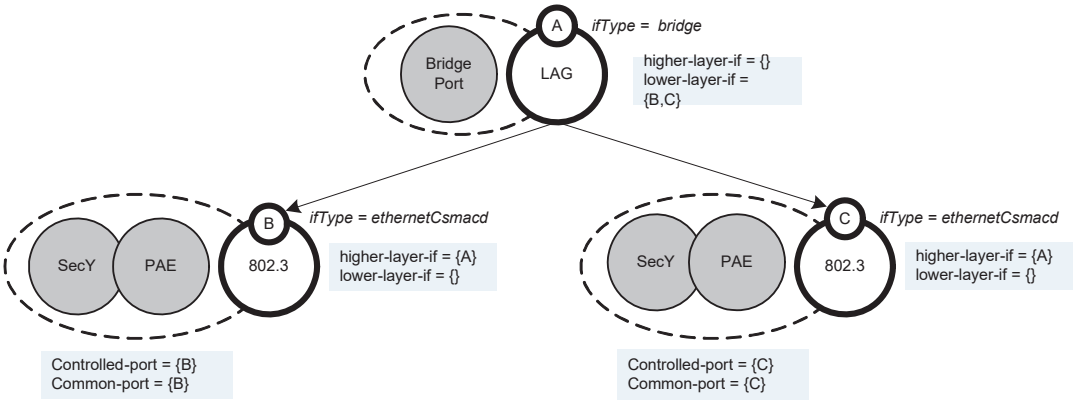


Figure 14-12—Augmented Interface Model of Bridge Port LAG with MACsec on members

The YANG models (defined in 14.5.3) supports the various interface augmentation and interface stacking strategies. Interface stacking models ranging from explicitly modeling interfaces to augmentation of interfaces can be supported. The controller (e.g., NETCONF client) can dictate which interface stack model should be used. Consideration needs to be made to select the interface stack model that is right for the usage scenario. For example, starting with an augmented interface model of a Bridge Port, as illustrated in Figure 14-6, and transitioning to a Bridge Port LAG interface stack (with or without MACsec), as illustrated in Figure 14-7, requires the initial Bridge Port Interfaces to be torn down (e.g., deleted) and re-instantiated (e.g., created) to support the transition. This is not the case when starting from an explicit interface model of the Bridge Port (as illustrated in Figure 14-5).

14.5 Definition of the IEEE 802.1X YANG data model[26]

14.5.1 ieee802-dot1x YANG tree schema

```
module: ieee802-dot1x
  +--rw nid-group
     +--rw pae-nid-group* [nid]
        +--rw nid                       dot1x-types:pae-nid
        +--rw use-eap?                  enumeration
        +--rw unauth-allowed?           enumeration
        +--rw unsecure-allowed?         enumeration
        +--rw unauthenticated-access?   enumeration
        +--rw access-capabilities?      dot1x-types:pae-nid-capabilities
        +--ro kmd?                      dot1x-types:pae-kmd

  augment /sys:system:
    +--rw pae-system
       +--rw name?                    string
       +--rw system-access-control?   enumeration
       +--rw system-announcements?    enumeration
       +--ro eapol-protocol-version?  uint8
       +--ro mka-version?             uint8
       +--ro pae*                     if:interface-ref
  augment /if:interfaces/if:interface:
    +--rw pae
       +--rw pae-system?                -> /sys:system/dot1x:pae-system/name
       +--rw vp-enable?                 boolean
       +--rw port-capabilities
       |  +--rw supp?                  boolean
       |  +--rw auth?                  boolean
       |  +--rw mka?                   boolean
       |  +--rw macsec?                boolean
       |  +--rw announcements?         boolean
       |  +--rw listener?              boolean
       |  +--rw virtual-ports?         boolean
       |  +--rw in-service-upgrades?   boolean
       +--ro port-name?                  if:interface-ref
       +--ro port-number?                dot1x-types:pae-if-index
       +--ro controlled-port-name?       if:interface-ref
       +--ro controlled-port-number?     dot1x-types:pae-if-index
       +--ro uncontrolled-port-name?     if:interface-ref
       +--ro uncontrolled-port-number?   dot1x-types:pae-if-index
       +--ro common-port-name?           if:interface-ref
       +--ro common-port-number?         dot1x-types:pae-if-index
       +--rw port-type?                  enumeration
       +--ro virtual-port
       |  +--ro max?            uint32
       |  +--ro current?        yang:gauge32
       |  +--ro start?          boolean
       |  +--ro peer-address?   ieee:mac-address
       +--rw supplicant
       |  +--rw held-period?     uint16
       |  +--rw retry-max?       uint32
       |  +--ro enabled?         boolean
       |  +--ro authenticate?    boolean
       |  +--ro authenticated?   boolean
       |  +--ro failed?          boolean
       +--rw authenticator
       |  +--rw quiet-period?    uint16
       |  +--rw reauth-period?   uint32
       |  +--rw reauth-enable?   boolean
       |  +--rw retry-max?       uint32
       |  +--ro enabled?         boolean
       |  +--ro authenticate?    boolean
       |  +--ro authenticated?   boolean
       |  +--ro failed?          boolean
```

[26] Copyright release for YANG: Users of this standard may freely reproduce the YANG modules contained in this standard so that they can be used for their intended purpose.

```
+--rw kay
|  +--rw enable?               boolean
|  +--rw actor
|  |  +--rw priority?   uint8
|  |  +--ro sci?        dot1x-types:sci-list-entry
|  +--rw key-server
|  |  +--rw priority?   uint8
|  |  +--ro sci?        dot1x-types:sci-list-entry
|  +--rw group
|  |  +--rw join?   boolean
|  |  +--rw form?   boolean
|  |  +--rw new?    boolean
|  +--rw macsec
|  |  +--rw capable?          boolean
|  |  +--rw desired?          boolean
|  |  +--ro protect?          boolean
|  |  +--ro validate?         boolean
|  |  +--ro replay-protect?   boolean
|  +--rw suspend-on-request?   boolean
|  +--rw suspend-for?          uint8
|  +--ro suspended-while?      uint8
|  +--ro active?               boolean
|  +--ro authenticated?        boolean
|  +--ro secured?              boolean
|  +--ro failed?               boolean
|  +--ro key-number
|  |  +--ro tx?   dot1x-types:mka-kn
|  |  +--ro rx?   dot1x-types:mka-kn
|  +--ro association-number
|  |  +--ro tx?   dot1x-types:mka-an
|  |  +--ro rx?   dot1x-types:mka-an
|  +--rw participants* [participant]
|     +--rw participant    uint32
|     +--rw cached?        boolean
|     +--rw active?        boolean
|     +--rw retain?        boolean
|     +--rw activate?      enumeration
|     +--ro peers
|     |  +--ro live*        dot1x-types:sci-list-entry
|     |  +--ro potential*   dot1x-types:sci-list-entry
|     +--ro ckn?           dot1x-types:pae-ckn
|     +--ro kmd?           dot1x-types:pae-kmd
|     +--ro nid?           dot1x-types:pae-nid
|     +--ro auth-data?     dot1x-types:pae-auth-data
|     +--ro principal?     boolean
|     +--ro dist-ckn?      dot1x-types:pae-ckn
+--rw logon-nid
|  +--rw selected?        dot1x-types:pae-nid
|  +--rw pae-nid-group* [nid]
|  |  +--rw nid                       dot1x-types:pae-nid
|  |  +--rw use-eap?                  enumeration
|  |  +--rw unauth-allowed?           enumeration
|  |  +--rw unsecure-allowed?         enumeration
|  |  +--rw unauthenticated-access?   enumeration
|  |  +--rw access-capabilities?      dot1x-types:pae-nid-capabilities
|  |  +--ro kmd?                      dot1x-types:pae-kmd
|  +--ro connected?       dot1x-types:pae-nid
|  +--ro requested?       dot1x-types:pae-nid
+--rw announcer
|  +--rw enable?     boolean
|  +--rw announce* [announces]
|     +--rw announces        uint32
|     +--rw pae-nid-group* [nid]
|     |  +--rw nid                       dot1x-types:pae-nid
|     |  +--rw use-eap?                  enumeration
|     |  +--rw unauth-allowed?           enumeration
|     |  +--rw unsecure-allowed?         enumeration
|     |  +--rw unauthenticated-access?   enumeration
|     |  +--rw access-capabilities?      dot1x-types:pae-nid-capabilities
|     |  +--ro kmd?                      dot1x-types:pae-kmd
|     +--ro nid?             dot1x-types:pae-nid
|     +--ro access-status?   dot1x-types:pae-access-status
```

```
+--rw listener
|  +--rw enable?         boolean
|  +--ro announcement* [announcements]
|     +--ro announcements             uint32
|     +--ro nid?                      dot1x-types:pae-nid
|     +--ro kmd?                      dot1x-types:pae-kmd
|     +--ro specific?                 boolean
|     +--ro access-status?            dot1x-types:pae-access-status
|     +--ro requested-nid?            boolean
|     +--ro unauthenticated-access?   dot1x-types:pae-access-status
|     +--ro access-capabilities?      dot1x-types:pae-nid-capabilities
|     +--ro cipher-suites* [index]
|        +--ro index                    uint16
|        +--ro cipherSuite?             string
|        +--ro cipherSuiteCapability?   uint32
+--ro eapol-statistics
|  +--ro invalid-eapol-frame-rx?       yang:counter32
|  +--ro eap-length-error-frames-rx?   yang:counter32
|  +--ro eapol-announcements-rx?       yang:counter32
|  +--ro eapol-announce-reqs-rx?       yang:counter32
|  +--ro eapol-port-unavailable?       yang:counter32
|  +--ro eapol-start-frames-rx?        yang:counter32
|  +--ro eapol-eap-frames-rx?          yang:counter32
|  +--ro eapol-logoff-frames-rx?       yang:counter32
|  +--ro eapol-mk-no-cfn?              yang:counter32
|  +--ro eapol-mk-invalid-frames-rx?   yang:counter32
|  +--ro last-eapol-frame-source?      ieee:mac-address
|  +--ro last-eapol-frame-version?     uint8
|  +--ro eapol-supp-eap-frames-tx?     yang:counter32
|  +--ro eapol-logoff-frames-tx?       yang:counter32
|  +--ro eapol-announcements-tx?       yang:counter32
|  +--ro eapol-announce-reqs-tx?       yang:counter32
|  +--ro eapol-start-frames-tx?        yang:counter32
|  +--ro eapol-auth-eap-frames-tx?     yang:counter32
|  +--ro eapol-mka-frames-tx?          yang:counter32
+--rw logon-process
   +--rw logon?                boolean
   +--ro connect?              enumeration
   +--ro port-valid?           boolean
   +--ro session-statistics* [session-id]
      +--ro session-id         dot1x-types:pae-session-id
      +--ro user-name?         dot1x-types:pae-session-user-name
      +--ro octets-rx?         yang:counter64
      +--ro octets-tx?         yang:counter64
      +--ro frames-rx?         yang:counter64
      +--ro frames-tx?         yang:counter64
      +--ro time?              uint32
      +--ro terminate-cause?   enumeration
```

14.5.2 ieee802-dot1x-types YANG module

```
module ieee802-dot1x-types {

  namespace "urn:ieee:std:802.1X:yang:ieee802-dot1x-types";
  prefix "dot1x-types";

  organization
    "Institute of Electrical and Electronics Engineers";

  contact
    "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@LISTSERV.IEEE.ORG";

  description
    "Port-based network access control allows a network administrator
    to restrict the use of IEEE 802 LAN service access points (ports)
    to secure communication between authenticated and authorized
    devices. IEEE Std 802.1X specifies an architecture, functional
    elements, and protocols that support mutual authentication
    between the clients of ports attached to the same LAN and secure
    communication between the ports. The following control allows a
    port to be reinitialized, terminating (and potentially
    restarting) authentication exchanges and MKA operation, based on
    a data model described in a set of YANG modules.";

  revision 2020-02-18 {
    description
      "Updated Contact information.";
  }

  revision 2019-05-28 {
    description
      "Updates based upon comment resolution on draft
      D1.0 of P802.1X-Rev.";
    reference
      "IEEE Std 802.1X-2020, Port-Based Network Access Control.";
  }

  /* ---------------------------------------------
   * Type definitions used by dot1X YANG module
   * ---------------------------------------------
   */

  typedef pae-nid {
    type string {
      length "0..100";
    }
    description
      "Network Identity, which is a UTF-8 string identifying a
      network or network service.";
    reference
      "IEEE 802.1X-2020 Clause 3, Clause 10.1, Clause 12.6";
  }

  typedef pae-session-user-name {
    type string {
      length "0..253";
    }
    description
      "Session user name, which is a UTF-8 string, representing the
```

```
    identity of the peer Supplicant.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}

typedef pae-session-id {
  type string {
    length "3..253";
  }
  description
    "Session Identifier, which is a UTF-8 string, uniquely
    identifying the session within the context of the PAE's
    system.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}

typedef pae-nid-capabilities {
  type bits {
    bit eap {
      position 0;
      description
        "EAP";
    }
    bit eapMka {
      position 1;
      description
        "EAP + MKA";
    }
    bit eapMkaMacSec {
      position 2;
      description
        "EAP + MKA + MACsec";
    }
    bit mka {
      position 3;
      description
        "MKA";
    }
    bit mkaMacSec {
      position 4;
      description
        "MKA + MACsec";
    }
    bit higherLayer {
      position 5;
      description
        "Higher Layer (WebAuth)";
    }
    bit higherLayerFallback {
      position 6;
      description
        "Higher Layer Fallback (WebAuth)";
    }
    bit vendorSpecific {
      position 7;
      description
        "Vendor specific authentication mechanisms";
    }
  }
  description
    "Authentication and protection capabilities supported for the
    NID. Indicates the combinations of authentication and
    protection capabilities supported for the NID. Any set of these
    combinations can be supported.";
  reference
    "IEEE 802.1X-2020 Clause 10.1, Clause 11.12.3";
}

typedef pae-access-status {
  type enumeration {
    enum no-access {
```

```
      description
        "Other than to authentication services, and to services
        announced as available in the absence of authentication
        (unauthenticated).";
    }
    enum remedial-access {
      description
        "The access granted is severely limited, possibly to
        remedial services.";
    }
    enum restricted-access {
      description
        "The Controlled Port is operational, but restrictions have
        been applied by the network that can limit access to some
        resources.";
    }
    enum expected-access {
      description
        "The Controlled Port is operational, and access provided is
        as expected for successful authentication and authorization
        for the NID.";
    }
  }
  description
    "Indicates the transmitter's Controlled Port operational status
    and current level of access resulting from authentication and
    the consequent authorization controls applied by that port's
    clients.";
  reference
    "IEEE 802.1X-2020 Clause 10.4, Clause 12.5";
}

typedef mka-kn {
  type uint32;
  description
    "Indicates a Key Number (KN) used in MKA. It is assigned by
    the Key Server (sequentially beginning with 1).";
  reference
    "IEEE 802.1X-2020 Clause 9.8, Clause 9.16";
}

typedef mka-an {
  type uint32;
  description
    "A number that is concatenated with a MACsec Secure Channel
    Identifier to identify a Secure Association. Indicates an
    Association Number (AN) assigned by the Key Server for use with
    the key number for transmission.";
  reference
    "IEEE 802.1X-2020 Clause 9.8, Clause 9.16";
}

typedef pae-ckn {
  type string {
    length "1..32";
  }
  description
    "Indicates the CAK name to identify the Connectivity
    Association Key (CAK) which is the root key in the MACsec Key
    Agreement key hierarchy. All potential members of the CA use
    the same CKN.";
  reference
    "IEEE 802.1X-2020 Clause 9.3.1, Clause 6.2";
}

typedef pae-kmd {
  type string {
    length "0..253";
  }
  description
    "A Key Management Domain (KMD). A string of up to 253 UTF-8
    characters that names the transmitting authenticator's key
```

```
      management domain.";
    reference
      "IEEE Clause 12.6";
  }

  typedef pae-auth-data {
    type string;
    description
      "Authorization data associated with the CAK.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }

  typedef sci-list-entry {
    type string {
      length "8";
    }
    description
      "8 octet string, where the first 6 octets represents the MAC
      Address (in canonical format), and the next 2 octets represents
      the Port Identifier.";
    reference
      "IEEE 802.1AE Clause 7.1.2, Clause 10.7.1";
  }

  typedef pae-if-index {
    type int32 {
      range "1..2147483647";
    }
    description
        "The interface index value represented by this interface.";
  }

} // ieee802-dot1x-types
```

14.5.3 ieee802-dot1x YANG module definition

```
module ieee802-dot1x {

  namespace "urn:ieee:std:802.1X:yang:ieee802-dot1x";
  prefix "dot1x";

  import ieee802-types { prefix "ieee"; }
  import ietf-yang-types { prefix "yang"; }
  import ietf-interfaces { prefix "if"; }
  import ietf-system { prefix "sys"; }
  import iana-if-type { prefix "ianaift"; }
  import ieee802-dot1x-types { prefix "dot1x-types"; }

  organization
    "Institute of Electrical and Electronics Engineers";

  contact
    "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@LISTSERV.IEEE.ORG";

  description
    "Port-based network access control allows a network administrator
    to restrict the use of IEEE 802 LAN service access points (ports)
    to secure communication between authenticated and authorized
    devices. IEEE Std 802.1X specifies an architecture, functional
    elements, and protocols that support mutual authentication
    between the clients of ports attached to the same LAN and secure
    communication between the ports. The following control allows a
    port to be reinitialized, terminating (and potentially
    restarting) authentication exchanges and MKA operation, based on
    a data model described in a set of YANG modules.";

  revision 2020-02-18 {
    description
      "Updated Contact information.";
  }

  revision 2019-06-12 {
    description
      "Updates based on comment resolution of the WG ballot of
      P802.1X-Rev/D1.0.";
    reference
      "IEEE Std 802.1X-2020, Port-Based Network Access Control.";
  }

  grouping nid-group {
    description
      "The PAE NID Group configuration and operational information.";
    list pae-nid-group {
      key "nid";
      description
        "A list that contains the configuration and operational
        nodes for the network announcement information for the
        Logon Process.";
      leaf nid {
        type dot1x-types:pae-nid;
        description
          "Identification of the network or network service.";
        reference
          "IEEE 802.1X-2020 Clause 12.5";
```

```
}
leaf use-eap {
  type enumeration {
    enum never {
      description
        "Never.";
    }
    enum immediate {
      description
        "Immediately, concurrently with the use of MKA with any
        cached CAK(s).";
    }
    enum mka-fail {
      description
        "Not until MKA has failed, if a prior CAK has been
        cached.";
    }
  }
  default "immediate";
  description
    "Determines when the Logon Process will initiate EAP, if
    the Supplicant and or Authenticator are enabled, and takes
    one of the above values.";
  reference
    "IEEE 802.1X-2020 Clause 12.5";
}
leaf unauth-allowed {
  type enumeration {
    enum never {
      description
        "Never.";
    }
    enum immediate {
      description
        "Immediately, independently of any current or future
        attempts to authenticate using the PAE or MKA.";
    }
    enum auth-fail {
      description
        "Not until an attempt has been made to authenticate
        using EAP, unless neither the supplicant nor the
        authenticator is enabled, and MKA has attempted to use
        any cached CAK (unless the KaY is not enabled).";
    }
  }
  default "immediate";
  description
    "Determines when the Logon Process will tell the CP state
    machine to provide unauthenticated connectivity, and takes
    one of the above values.";
  reference
    "IEEE 802.1X-2020 Clause 12.5";
}
leaf unsecure-allowed {
  type enumeration {
    enum never {
      description
        "Never.";
    }
    enum immediate {
      description
        "Immediately, to provide connectivity concurrently with
        the use of MKA with any CAK acquired through EAP.";
    }
    enum mka-fail {
      description
        "Not until MKA has failed, or is not enabled.";
    }
    enum mka-server {
      description
        "Only if directed by the MKA server.";
    }
```

```
      }
      default "immediate";
      description
        "Determines when the Logon Process will tell the CP state
        machine to provide authenticated but unsecured
        connectivity, takes one of the above values.";
      reference
        "IEEE 802.1X-2020 Clause 12.5";
    }
    leaf unauthenticated-access {
      type enumeration {
        enum no-access {
          description
            "Other than to authentication services.";
        }
        enum fallback-access {
          description
            "Limited access can be provided after authentication
            failure.";
        }
        enum limited-access {
          description
            "Immediate limited access is available without
            authentication.";
        }
        enum open-access {
          description
            "Immediate access is available without
            authentication.";
        }
      }
      default "no-access";
      description
        "Unauthenticated access capabilities provided by the NID.";
      reference
        "IEEE 802.1X-2020 Clause 10.1";
    }
    leaf access-capabilities {
      type dot1x-types:pae-nid-capabilities;
      description
        "Authentication and protection capabilities supported for
        the NID.";
      reference
        "IEEE 802.1X-2020 Clause 10.1";
    }

    leaf kmd {
      type dot1x-types:pae-kmd;
      config false;
      description
        "The Key Management Domain for the NID.";
      reference
        "IEEE 802.1X-2020 Clause 10.4";
    }
  }
}

grouping port-capabilities {
  description
    "Per port PAE feature capabilities.";
  leaf supp {
    type boolean;
    description
      "Indicates if PACP EAP Supplicant is supported.";
    reference
      "IEEE 802.1X-2020 Clause 12.9.2";
  }
  leaf auth {
    type boolean;
    description
      "Indicates if PACP EAP Authenticator is supported.";
    reference
```

```
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf mka {
      type boolean;
      description
        "Indicates if MKA is supported.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf macsec {
      type boolean;
      description
        "Indicates if MACsec on the Controlled port is supported.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf announcements {
      type boolean;
      description
        "Indicates if the ability to send EAPOL announcements is
        supported.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf listener {
      type boolean;
      description
        "Indicates if the ability to use received EAPOL
        announcements is supported.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf virtual-ports {
      type boolean;
      description
        "Indicates if virtual ports for a real port is supported.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf in-service-upgrades {
      type boolean;
      description
        "Indicates if MKA in-service upgrades is supported.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
  }

  /* ---------------------------------------------------
   * Configuration objects used by 802.1X YANG module
   * ---------------------------------------------------
   */
  augment "/sys:system" {
    description
      "Augment system with 802.1X PAE System specific configuration
      nodes.";
    container pae-system {
      description
        "Contains all 802.1X PAE System specific related
        configuration and operational data.";
      leaf name {
        type string {
          length "1..255";
        }
        description
          "The name which uniquely identifies the PAE System.";
      }
      leaf system-access-control {
        type enumeration {
          enum disabled {
            description
              "Deletes any virtual ports previously instantiated, and
```

```
            terminates authentication exchanges and MKA
            operation.";
        }
        enum enabled {
          description
            "Enables PAE system access control.";
        }
      }
      description
        "Setting this control to disabled deletes any virtual ports
        previously instantiated, and terminates authentication
        exchanges and MKA operation. Each real port PAE behaves as
        if enabledVirtualPorts was clear, the PAEs Supplicant,
        Authenticator, and KaY as if their enabled controls were
        clear, and Logon Process(es) as if unauthAllowed was
        Immediate. Announcements can be transmitted (subject to
        other controls), both periodically and in response to
        announcement requests (conveyed by EAPOL-Starts or
        EAPOL-Announcement-Reqs) but are sent with a single NID
        Set, with a null NID, and the Access Information TLV (and
        no other) with an pae-access-status of No Access,
        accessRequested false, OpenAccess, and no
        accessCapabilities. The control variable settings for each
        real port PAE are unaffected, and will be used once
        systemAccessControl is set to enabled.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.1";
    }
    leaf system-announcements {
      type enumeration {
        enum disabled {
          description
            "Causes each PAE to behave as if enabled were clear
            for the PAE's Announcement functionality.";
        }
        enum enabled {
          description
            "Enables PAE system announcements.";
        }
      }
      description
        "Setting this control to Disabled causes each PAE to behave
        as if enabled were clear for the PAE's Announcement
        functionality. The independent controls for each PAE apply
        if systemAnnouncements is Enabled.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.1";
    }
    leaf eapol-protocol-version {
      type uint8;
      config false;
      description
        "The EAPOL protocol version for this system.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.1, Clause 11.3";
    }
    leaf mka-version {
      type uint8;
      config false;
      description
        "The MKA protocol version for this system.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.1, Clause 11.3";
    }
    leaf-list pae {
      type if:interface-ref;
      config false;
      description
        "List of PAE references.";
    }
  }
}
```

```
/*
 *  Port Authentication Entity (PAE) Nodes
 */
augment "/if:interfaces/if:interface" {
  when "if:type = 'ianaift:ethernetCsmacd' or
        if:type = 'ianaift:ilan' or
        if:type = 'ianaift:macSecControlledIF' or
        if:type = 'ianaift:ptm' or
        if:type = 'ianaift:bridge'" {
    description
      "Applies to the Controlled Port of SecY or PAC shim or
      Ethernet related Interface.";
  }
  description
    "Augment interface model with PAE configuration and
      operational nodes.";
  reference
    "IEEE 802.1AE Clause 11.7 and IEEE 802.1X-2020 Clause 6.5 and
    Clause 13.3.2";
  container pae {
    description
      "Contains PAE configuration and operational related nodes.";
    leaf pae-system {
      type leafref {
        path "/sys:system/dot1x:pae-system/dot1x:name";
      }
      description
        "The PAE system that this PAE is a member of.";
    }
    leaf vp-enable {
      when "../port-type = 'real-port' and
            ../port-capabilities/virtual-ports = 'true'" {
        description
          "Applies when port is Real Port and virtual port
          capabilities are supported.";
      }
      type boolean;
      default "false";
      description
        "A real port's PAE may be configured to create virtual
        ports to support multi-access LANs provided that MKA and
        MACsec operation is enabled for that port.";
      reference
        "IEEE 802.1X-2020 Clause 12.7";
    }
    container port-capabilities {
      description
        "Per port PAE feature capabilities.";
      uses port-capabilities;
    }

    leaf port-name {
      type if:interface-ref;
      config false;
      description
        "Each PAE is uniquely identified by a port name.";
      }
    leaf port-number {
      type dot1x-types:pae-if-index;
      config false;
      description
        "Each PAE is uniquely identified by a port number. The
        port number used is unique amongst all port names for the
        system, and directly or indirectly identifies the
        Uncontrolled Port that supports the PAE. If the PAE has
        been dynamically instantiated to support an existing or
        potential virtual port, this portNumber, the
        uncontrolledPortNumber and the controlledPortNumber are
        allocated by the real ports PAE, and this portNumber is the
        uncontrolledPortNumber. If the PAE supports a real port,
        this portNumber is the commonPortNumber for the associated
```

```
    PAC or SecY.";
  reference
    "IEEE 802.1X-2020 Clause 12.9.2";
  }
leaf controlled-port-name {
  type if:interface-ref;
  config false;
  description
    "Each PAE is uniquely identified by a port name.";
}
leaf controlled-port-number {
  type dot1x-types:pae-if-index;
  config false;
  description
    "The port for the associated PAC or SecYs Controlled
    Port.";
  reference
    "IEEE 802.1X-2020 Clause 12.9.2";
}
leaf uncontrolled-port-name {
  type if:interface-ref;
  config false;
  description
    "The uncontrolled port name reference.";
}
leaf uncontrolled-port-number {
  type dot1x-types:pae-if-index;
  config false;
  description
    "The port for the associated PAC or SecYs Uncontrolled
    Port.";
  reference
    "IEEE 802.1X-2020 Clause 12.9.2";
}
leaf common-port-name {
  type if:interface-ref;
  config false;
  description
    "The common port name reference.";
}
leaf common-port-number {
  type dot1x-types:pae-if-index;
  config false;
  description
    "The port for the associated PAC or SecYs Common Port. All
    the virtual ports created for a given real port share the
    same Common Port and commonPortNumber.";
  reference
    "IEEE 802.1X-2020 Clause 12.9.2";
}
leaf port-type {
  type enumeration {
    enum real-port {
      description
        "Real Port type.";
    }
    enum virtual-port {
      description
        "Virtual Port type.";
    }
  }
  //config false;
  description
    "The port type of the PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.9.2";
}
container virtual-port {
  when "../port-capabilities/virtual-ports = 'true'" {
    description
      "Applies when the virtual ports port capability is
      supported.";
```

```
    }
    config false;
    description
      "Contains Virtual Port operational state information.";
    leaf max {
      when "../../port-type = 'real-port'" {
        description
          "Applies when Port is a Real Port.";
      }
      type uint32;
      description
        "The guaranteed maximum number of virtual ports.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf current {
      when "../../port-type = 'real-port'" {
        description
          "Applies when Port is a Real Port.";
      }
      type yang:gauge32;
      description
        "The current number of virtual ports.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.2";
    }
    leaf start {
      when "../../port-type = 'virtual-port'" {
        description
          "Applies when Port is a Virtual Port.";
      }
      type boolean;
      description
        "Set if the virtual port was created by receipt of an
        EAPOL-Start frame.";
      reference
        "IEEE 802.1X-2020 Clause 12.9.7";
    }
    leaf peer-address {
      when "../../port-type = 'virtual-port'" {
        description
          "Applies when Port is a Virtual Port.";
      }
      type ieee:mac-address;
      description
        "The source MAC Address of the EAPOL-Start (if vpStart is
        set).";
      reference
        "IEEE 802.1X-2020 Clause 12.9.7";
    }
  }

  container supplicant {
    when "../port-type = 'real-port' and
          ../port-capabilities/supp = 'true'" {
      description
        "Applies to Real Port when supplicant port capabilities
        are supported.";
    }
    description
      "Contains the configuration nodes for the Supplicant PAE
      associated with each port.";
    leaf held-period {
      type uint16;
      units seconds;
      default "60";
      description
        "The initial value of the timer used to impose a wait
        period after a failed authentication attempt, before
        another attempt is permitted.";
      reference
        "IEEE 802.1X-2020 Clause 8.6";
```

```
    }
    leaf retry-max {
      type uint32;
      default "2";
      description
        "Specifies the maximum number of re-authentication
        attempts on an authenticator port before port is
        unauthorized.";
      reference
        "IEEE 802.1X-2020 Clause 8.7";
    }

    leaf enabled {
      type boolean;
      config false;
      description
        "Set by PACP if the PAE can provide authentication. Will
        be FALSE if the Port is not enabled, if the functionality
        provided by the PAE is not available, or not implemented,
        or the control variable enable has been cleared by
        management, e.g. because the application scenario
        authenticates a user and there is no user logged on.";
      reference
        "IEEE 802.1X-2020 Clause 8.4";
    }
    leaf authenticate {
      type boolean;
      config false;
      description
        "Set by the PAE client to request authentication, and
        allows reauthentication while set. Cleared by the client
        to revoke authentication. To enable authentication the
        client also needs to clear failed (if set).";
      reference
        "IEEE 802.1X-2020 Clause 8.4";
    }
    leaf authenticated {
      type boolean;
      config false;
      description
        "Set by PACP if the PAE is currently authenticated, and
        cleared if the authentication fails or is revoked.";
      reference
        "IEEE 802.1X-2020 Clause 8.4";
    }
    leaf failed {
      type boolean;
      config false;
      description
        "Set by PACP if the authentication has failed or has been
        terminated. The cause could be a Fail returned by EAP,
        either immediately or following a reauthentication, an
        excessive number of attempts to authenticate (either
        immediately or upon reauthentication), or the client
        deasserting authenticate. The PACP will clear
        authenticated as well as setting failed. Any ongoing
        authentication exchange will be terminated (by the state
        machines) if enable becomes FALSE and enabled will be
        cleared, but failed will not be set.";
      reference
        "IEEE 802.1X-2020 Clause 8.4";
    }
  }

  container authenticator {
    when "../port-capabilities/auth = 'true'" {
      description
        "Applies when the Authenticator is supported.";
    }
    description
      "Contains configuration nodes for the Authenticator PAE
      associated with each port.";
```

```
leaf quiet-period {
  type uint16;
  units seconds;
  default "60";
  description
    "Number of seconds that the authenticator remains in the quiet
    state following a failed authentication exchange with the
    supplicant.";
  reference
    "IEEE 802.1X-2020 Clause 8.6, Figure 12-3";
}
leaf reauth-period {
  type uint32;
  units seconds;
  default "3600";
  description
    "This object indicates the time period of the
    reauthentication to the supplicant.";
  reference
    "IEEE 802.1X-2020 Clause 8.6, Figure 12-3";
}
leaf reauth-enable {
  type boolean;
  default "false";
  description
    "Re-authentication is enabled or not.";
  reference
    "IEEE 802.1X-2020 Clause 5.8 and 8.9";
}
leaf retry-max {
  type uint32;
  default "2";
  description
    "Specifies the maximum number of re-authentication
    attempts on an authenticator port before port is
    unauthorized.";
  reference
    "IEEE 802.1X-2020 Clause 8.9";
}

leaf enabled {
  type boolean;
  config false;
  description
    "Set by PACP if the PAE can provide authentication. Will
    be FALSE if the Port is not enabled, if the functionality
    provided by the PAE is not available, or not implemented,
    or the control variable enable has been cleared by
    management, e.g. because the application scenario
    authenticates a user and there is no user logged on.";
  reference
    "IEEE 802.1X-2020 Clause 8.4";
}
leaf authenticate {
  type boolean;
  config false;
  description
    "Set by the PAE client to request authentication, and
    allows reauthentication while set. Cleared by the client
    to revoke authentication. To enable authentication the
    client also needs to clear failed (if set).";
  reference
    "IEEE 802.1X-2020 Clause 8.4";
}
leaf authenticated {
  type boolean;
  config false;
  description
    "Set by PACP if the PAE is currently authenticated, and
    cleared if the authentication fails or is revoked.";
  reference
    "IEEE 802.1X-2020 Clause 8.4";
```

```
    }
    leaf failed {
      type boolean;
      config false;
      description
        "Set by PACP if the authentication has failed or has been
        terminated. The cause could be a Fail returned by EAP,
        either immediately or following a reauthentication, an
        excessive number of attempts to authenticate (either
        immediately or upon reauthentication), or the client
        deasserting authenticate. The PACP will clear
        authenticated as well as setting failed. Any ongoing
        authentication exchange will be terminated (by the state
        machines) if enable becomes FALSE and enabled will be
        cleared, but failed will not be set.";
      reference
        "IEEE 802.1X-2020 Clause 8.4";
    }
  }

  container kay {
    when "../port-capabilities/mka = 'true'" {
      description
        "Applies when the MKA port capability is supported.";
    }
    description
      "Contains configuration system level information for each
      Interface supported by the KaY (Key Aggreement Entity).";
    leaf enable {
      type boolean;
      default "false";
      description
        "Set by management to enable (clear to disable) the use
        of MKA.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    container actor {
      description
        "Contains configuration and operational nodes
        associated with the actor";
      leaf priority {
        type uint8;
        description
          "The Key Server Priority for all the ports actors.";
        reference
          "IEEE 802.1X-2020 Clause 9.16";
      }
      leaf sci {
        type dot1x-types:sci-list-entry;
        config false;
        description
          "The SCI assigned by the system to the port (applies
          to all the ports actors).";
        reference
          "IEEE 802.1X-2020 Clause 9.16";
      }
    }
    container key-server {
      description
        "Contains configuration and operational nodes
        associated with the key
        server.";
      leaf priority {
        type uint8;
        description
          "The Key Server Priority for the Key Server for the
          principal actor. Matches the actorPriority if the
          actor is the Key Server";
        reference
          "IEEE 802.1X-2020 Clause 9.16";
      }
```

```
    leaf sci {
      type dot1x-types:sci-list-entry;
      config false;
      description
        "The SCI for Key Server for the principal actor. Null
        if there is no principal actor, or that actor has no
        live peers. Matches the actorSCI if the actor is the
        Key Server.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
  }
  container group {
    description
      "Contains configuration nodes associated with the
      group.";
    leaf join {
      type boolean;
      default "true";
      description
        "Set if the KaY will accept Group CAKs distributed by
        MKA.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    leaf form {
      type boolean;
      default "false";
      description
        "Set if the KaY will attempt to use point-to-point CAs
        to distribute a Group CAK, if its principal actor is
        the Key Server for all the point-to-point CAs.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    leaf new {
      type boolean;
      default "false";
      description
        "Set by management if a new Group CAK is to be
        distributed, if the principal actor is the Key Server
        for all point-to-point CAs. Cleared by the KaY when
        distribution is complete.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
  }

  container macsec {
    when "../../port-capabilities/macsec = 'true'" {
      description
        "Applies when the MACsec port capability is
        supported.";
    }
    description
      "Contains configuration and operational nodes
      associated with macsec.";
    leaf capable {
      type boolean;
      description
        "Set if MACsec is implemented.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    leaf desired {
      type boolean;
      default "true";
      description
        "Set if the participant desires MACsec frame protection.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
```

```
    leaf protect {
      type boolean;
      config false;
      description
        "As used by the CP state machine, see 12.4.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    leaf validate {
      type boolean;
      config false;
      description
        "As used by the CP state machine, see 12.4.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    leaf replay-protect {
      type boolean;
      config false;
      description
        "As used by the CP state machine, see 12.4.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
  }
  leaf suspend-on-request {
    type boolean;
    default "true";
    description
      "Set by management to allow the KaYs principal actor to
      initiate a suspension if it is the Key Server and another
      participant has requested a suspension.";
  }
  leaf suspend-for {
    type uint8;
    default "0";
    description
      "Set by management to a non-zero number of seconds
      between 1 and MKA Suspension Limit to initiate a
      suspension (9.18) of that duration (if the KaYs principal
      actor is the Key Server) or to request a suspension
      (otherwise).";
    reference
      "IEEE 802.1X-2020 Clause 9.18";
  }

  leaf suspended-while {
    type uint8;
    config false;
    description
      "Read by management to determine if a suspension is in
      progress and (when available) to discover the remaining
      duration of that suspension";
    reference
      "IEEE 802.1X-2020 Clause 9.18";
  }
  leaf active {
    type boolean;
    config false;
    description
      "Set if there is at least one active actor, transmitting
      MKPDUs.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
  leaf authenticated {
    type boolean;
    config false;
    description
      "Set if the principal actor, i.e. the participant that
      has the highest priority Key Server and one or more live
```

```
    peers, has determined that Controlled Port communication
    should proceed without MACsec.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
leaf secured {
  type boolean;
  config false;
  description
    "Set if the principal actor has determined that
    communication should use MACsec.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
leaf failed {
  type boolean;
  config false;
  description
    "Cleared when authenticated or secured are set, set if
    the latter are clear and MKA Life Time has elapsed since
    an MKA participant was last created.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
container key-number {
    config false;
  description
    "Contains operation state nodes for Key Numbers.";
  leaf tx {
    type dot1x-types:mka-kn;
    description
      "The Key Number assigned by the Key Server to the SAK
      currently being used for transmission. Null if MACsec
      is not being used.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
  leaf rx {
    type dot1x-types:mka-kn;
    description
      "The Key Number assigned by the Key Server to the
      oldest SAK currently being used for reception. The same
      as txKN if a single SAK is currently in use (as will
      most often be the case). Null if MACsec is not being
      used.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
}
container association-number {
    config false;
  description
    "Contains operation state nodes for Association
    Numbers.";
  leaf tx {
    type dot1x-types:mka-an;
    description
      "The Association Number assigned by the Key Server for
      use with txKN. Zero if MACsec is not in use.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
  leaf rx {
    type dot1x-types:mka-an;
    description
      "The Association Number assigned by the Key Server for
      use with rxKN. The same as txAN if a single SAK is
      currently in use. Zero if MACsec is not in use.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
}
```

```
list participants {
  key "participant";
  description
    "Contains list of configuration and operational nodes
    for each MKA participant supported by the KaY MKA
    entity.";
  leaf participant {
    type uint32;
    description
      "Key into Participants list.";
  }
  leaf cached {
    type boolean;
    description
      "Set by the KaY if the participants parameters are
      cached. If set, cached can be cleared by management to
      remove the participant from the cache.";
  }
  leaf active {
    type boolean;
    default "false";
    description
      "Set if the participant is active, i.e., is currently
      transmitting periodic MKPDUs.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
  leaf retain {
    type boolean;
    default "false";
    description
      "Set by management to retain the participant in the
      cache, even if the KaY would normally remove it (due to
      lack of use for example).";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
  leaf activate {
    type enumeration {
      enum default {
        description
          "The participant is from cached entries created by
          the KaY as part of normal operation, without
          explicit management, and is activated according to
          the implementation dependent policies of the KaY.";
      }
      enum disabled {
        description
          "The participant allows the cache information to be
          retained, but disabled for indefinite period.";
          }
      enum on-oper-up {
        description
          "Causing the participant to be activated when the
          PAEs part is activated, and therefore when the SecY
          or PACs Common Port becomes operational.";
      }
      enum always {
        description
          "Causing the participant to remain active all the
          time, even in the continued absence of partners.";
      }
    }
    default "default";
    description
      "Controls when the participant is activated. Cached
      entries created by the KaY as part of normal operation,
      without explicit management, have the value Default,
      and are activated according to the implementation
      dependent policies of the KaY. This variable can be
      set to any of its values by management. Disabled allows
```

```
    the cache entry to be retained, but disabled for an
    indefinite period. OnOperUp causes the participant to
    be activated when the PAEs port (and therefore when the
    SecY or PACs Common Port becomes MAC_Operational).
    Always causes the participant to remain active all the
    time, even in the continued absence of partners. If the
    value is changed to Disabled or OnOperUp, the
    participant ceases operation immediately and receipt of
    MKPDUs with a matching CKN during a subsequent period
    of twice MKA Life Time will not cause the participant
    to become active once more.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}

container peers {
  config false;
  description
    "Contains operational state nodes associated with the
    Peers.";
  leaf-list live {
    type dot1x-types:sci-list-entry;
      description
        "A list of the SCIs of the participants live
        peers.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
  }
  leaf-list potential {
    type dot1x-types:sci-list-entry;
    description
      "A list of the SCIs of the participants potential
      peers.";
    reference
      "IEEE 802.1X-2020 Clause 9.16";
  }
}
leaf ckn {
  type dot1x-types:pae-ckn;
  config false;
  description
    "The secure Connectivity Association Key Name for the
    participant.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
leaf kmd {
  type dot1x-types:pae-kmd;
  config false;
  description
    "The Key Management Domain for the participant.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
leaf nid {
  type dot1x-types:pae-nid;
  config false;
  description
    "The NID for the participant.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
leaf auth-data {
  type dot1x-types:pae-auth-data;
  config false;
  description
    "Authorization data associated with the secure
    Connectivity Association Key.";
  reference
    "IEEE 802.1X-2020 Clause 9.16";
}
leaf principal {
```

```
      type boolean;
      config false;
      description
        "Set if the participant is currently the principal
        actor.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
    leaf dist-ckn {
      type dot1x-types:pae-ckn;
      config false;
      description
        "The CKN for the last CAK distributed (either by the
        actor or one of its partners). Null if this participant
        has not been used to distribute a CAK.";
      reference
        "IEEE 802.1X-2020 Clause 9.16";
    }
  }
}

container logon-nid {
  description
    "Contains the configuration and operational related NID
      information for the Logon Process. The Logon Process may
      use Network Identifiers (NIDs) to manage its use of
      authentication credentials, cached CAKs, and
      announcements.";
  leaf selected {
    type dot1x-types:pae-nid;
    description
      "The NID currently configured for use by an access
      controlled port when transmitting EAPOL-Start frames.
      Defaults to the null NID.";
    reference
      "IEEE 802.1X-2020 Clause 12.5";
  }
  uses nid-group;

  leaf connected {
    type dot1x-types:pae-nid;
    config false;
    description
      "The NID associated with the current connectivity
      (possibly unauthenticated) provided by the operation of
      the CP state machine.";
    reference
      "IEEE 802.1X-2020 Clause 12.5";
  }
  leaf requested {
    type dot1x-types:pae-nid;
    config false;
    description
      "The NID marked as Access requested in announcements, as
      determined from EAPOL-Start frames. Defaults to the
      selectedNID.";
    reference
      "IEEE 802.1X-2020 Clause 12.5";
  }
}

container announcer {
  when "../port-capabilities/announcements = 'true'" {
    description
      "Applies when the Announcements port capabilities are
      supported.";
  }
  description
    "Contains the configuration related Announcer
    information.";
  leaf enable {
    type boolean;
```

```
    default "false";
    description
      "A boolean indicating if the announcer is enabled or
      not.";
    reference
      "IEEE 802.1X-2020 Clause 10.4";
  }
  list announce {
    key "announces";
    description
      "Contains the configuration related status information
      that the Announcers announce in the network announcement
      of the PAE system.";
    leaf announces {
      type uint32;
      description
        "Key into Announce list.";
    }
    uses nid-group;

    leaf nid {
      type dot1x-types:pae-nid;
      config false;
      description
        "The NID information to identify a received network
        announcement for the PAE.";
      reference
        "IEEE 802.1X-2020 Clause 10.4";
    }
    leaf access-status {
      type dot1x-types:pae-access-status;
      config false;
      description
        "Access Status reflects connectivity as a result of
        authentication attempts, and might be set directly by
        the system or configured by AAA protocols.";
      reference
        "IEEE 802.1X-2020 Clause 10.4, Clause 12.5";
    }
  }
}

container listener {
  when "../port-capabilities/listener = 'true'" {
    description
      "Applies when the Listener port capability is
      supported.";
  }
  description
    "Contains the configuration and operational Listener
      node related information.";
  leaf enable {
    type boolean;
    default "false";
    description
      "A boolean indicating if the listener is enabled or
      not.";
    reference
      "IEEE 802.1X-2020 Clause 10.4";
  }

  list announcement {
    key "announcements";
    config false;
    description
      "A list containing the operational status information
      that the Listeners receive in the network announcement of
      the PAE system.";
    leaf announcements {
      type uint32;
      description
        "The key into the list of Announce nodes.";
```

```
}
leaf nid {
  type dot1x-types:pae-nid;
  description
    "The NID information to identify a received network
    announcement for the PAE.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
leaf kmd {
  type dot1x-types:pae-kmd;
  description
    "The KMD information for this received network
    announcement of the PAE.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
leaf specific {
  type boolean;
  description
    "This object indicates the received announcement
    information was specific to the receiving PAE, not
    generic for all systems attached to the LAN.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
leaf access-status {
  type dot1x-types:pae-access-status;
  description
    "The object information reflects connectivity as a
    result of authentication attempts for this received
    network announcement of the PAE.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
leaf requested-nid {
  type boolean;
  description
    "The authenticated access has been requested for this
    particular NID or not.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
leaf unauthenticated-access {
  type dot1x-types:pae-access-status;
  description
    "The access capability of the ports clients without
    authentication in this received network announcement of
    the PAE";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
leaf access-capabilities {
  type dot1x-types:pae-nid-capabilities;
  description
    "The authentication and protection capabilities
    supported for the NID.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
}
list cipher-suites {
  key "index";
  description
    "A table contains the Cipher Suites information that
    the Listeners receive in the network announcement of
    the PAE system.";
  reference
    "IEEE 802.1X-2020 Clause 10.4";
  leaf index {
    type uint16;
    description
      "Key into cipher suite entry.";
```

```
      }
      leaf cipherSuite {
        type string;
        description
          "cipher Suite identifier.";
      }
      leaf cipherSuiteCapability {
        type uint32;
        description
          "Cipher Suite capability.";
      }
    }
  }
}

container eapol-statistics {
  config false;
  description
    "Contains operational EAPOL statistics.";
  leaf invalid-eapol-frame-rx {
    when "../../port-type = 'real-port'" {
      description
        "Applies when port is Real Port.";
    }
    type yang:counter32;
    description
      "The number of invalid EAPOL frames of any type that
      have been received by this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eap-length-error-frames-rx {
    when "../../port-type = 'real-port'" {
      description
        "Applies when port is Real Port.";
    }
    type yang:counter32;
    description
      "The number of EAPOL frames that the Packet Body Length
      does not match a Packet Body that is contained within the
      octets of the received EAPOL MPDU in this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-announcements-rx {
    when "../../port-type = 'real-port'" {
      description
        "Applies when port is Real Port.";
    }
    type yang:counter32;
    description
      "The number of EAPOL-Announcement frames that have been
      received by this PAE";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-announce-reqs-rx {
    when "../../port-type = 'real-port'" {
      description
        "Applies when port is Real Port.";
    }
    type yang:counter32;
    description
      "The number of EAPOL-Announcement-Req frames that have
      been received by this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-port-unavailable {
    when "../../port-type = 'real-port' and
          ../../port-capabilities/virtual-ports = 'true'" {
      description
```

```
        "Applies when port is Real Port and when the virtual
        ports capability is supported.";
    }
    type yang:counter32;
    description
      "The number of EAPOL frames that are discarded because
      their processing would require the creation of a virtual
      port, for which there are inadequate or constrained
      resources, or an existing virtual port and no such port
      currently exists. If virtual port is not supported, this
      object should be always 0.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-start-frames-rx {
    type yang:counter32;
    description
      "The number of EAPOL-Start frames that have been received
      by this PAE";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-eap-frames-rx {
    type yang:counter32;
    description
      "The number of EAPOL-EAP frames that have been received
      by this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-logoff-frames-rx {
    type yang:counter32;
    description
      "The number of EAPOL-Logoff frames that have been
      received by this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-mk-no-cfn {
    type yang:counter32;
    description
      "The number of MKPDUs received with MKA not enabled or
      CKN not recognized in this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf eapol-mk-invalid-frames-rx {
    type yang:counter32;
    description
      "The number of MKPDUs failing in message authentication
      on receipt process in this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.1";
  }
  leaf last-eapol-frame-source {
    when "../../port-type = 'real-port'" {
      description
        "Applies when port is Real Port.";
    }
    type ieee:mac-address;
    description
      "The source MAC address of last received EAPOL frame by
      this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.2";
  }
  leaf last-eapol-frame-version {
    type uint8;
    description
      "The version of last received EAPOL frame by this PAE.";
    reference
      "IEEE 802.1X-2020 Clause 12.8.2";
```

```
}
leaf eapol-supp-eap-frames-tx {
  when "../../port-type = 'real-port'" {
    description
      "Applies when port is Real Port.";
  }
  type yang:counter32;
  description
    "The number of EAPOL-EAP frames that have been
    transmitted by the supplicant of this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
}
leaf eapol-logoff-frames-tx {
  when "../../port-type = 'real-port'" {
    description
      "Applies when port is Real Port.";
  }
  type yang:counter32;
  description
    "The number of EAPOL-Logoff frames that have been
    transmitted by this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
}
leaf eapol-announcements-tx {
  when "../../port-type = 'real-port'" {
    description
      "Applies when port is Real Port.";
  }
  type yang:counter32;
  description
    "The number of EAPOL-Announcement frames that have been
    transmitted by this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
}
leaf eapol-announce-reqs-tx {
  when "../../port-type = 'real-port'" {
    description
      "Applies when port is Real Port.";
  }
  type yang:counter32;
  description
    "The number of EAPOL-Announcement-Req frames that have
    been transmitted by this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
}
leaf eapol-start-frames-tx {
  type yang:counter32;
  description
    "The number of EAPOL-Start frames that have been
    transmitted by this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
}
leaf eapol-auth-eap-frames-tx {
  type yang:counter32;
  description
    "The number of EAPOL-EAP frames that have been
    transmitted by the authenticator of this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
}
leaf eapol-mka-frames-tx {
  type yang:counter32;
  description
    "The number of EAPOL-MKA frames with no CKN information
    that have been transmitted by this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.8.3";
```

```
    }
  }

  container logon-process {
    description
      "Contains configuration and operational system level
       information for each port to support the Logon Process(es)
       status information.";
    leaf logon {
      type boolean;
      default "false";
      description
        "A boolean indicating if the logon-process is enabled or
        not.";
      reference
        "IEEE 802.1X-2020 Clause 12.5";
    }

    leaf connect {
      type enumeration {
        enum pending {
          description
            "Prevent connectivity by clearing the
            controlledPortEnabled parameter.";
        }
        enum unauthenticated {
          description
            "Provide unsecured connectivity, setting
            controlledPortEnabled.";
        }
        enum authenticated {
          description
            "Provide unsecured connectivity with authorization
            data, setting controlledPortEnabled.";
        }
        enum secure {
          description
            "Provide secure connectivity, using SAKs provided by
            the KaY (when available) and setting
            controlledPortEnabled when those keys are installed
            and in use, as specified in detail by the CP state
            machine.";
        }
      }
      config false;
      description
        "The Logon Process sets this variable to one of the
        above values.";
      reference
        "IEEE 802.1X-2020 Clause 12.3";
    }
    leaf port-valid {
      type boolean;
      config false;
      description
        "Set if Controlled Port communication is secured as
        specified by the MACsec control macsecProtect.";
      reference
        "IEEE 802.1X-2020 Clause 12.3";
    }
    list session-statistics {
      key "session-id";
      config false;
      description
        "Contains operational state nodes associated with the
        session statistics.";
      leaf session-id {
        type dot1x-types:pae-session-id;
        description
          "Key into list of session statistics.";
        reference
          "IEEE 802.1X-2020 Clause 12.5.1";
```

```
}
leaf user-name {
  type dot1x-types:pae-session-user-name;
  description
    "User name of the session.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}
leaf octets-rx {
  type yang:counter64;
  description
    "The number of octets received in this session of this
    PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}
leaf octets-tx {
  type yang:counter64;
  description
    "The number of octets transmitted in this session of
    this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}
leaf frames-rx {
  type yang:counter64;
  description
    "The number of packets received in this session of
    this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}
leaf frames-tx {
  type yang:counter64;
  description
    "The number of packets transmitted in this session of
    this PAE.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}
leaf time {
  type uint32;
  units "seconds";
  description
    "Session Time. The duration of the session in
    seconds.";
  reference
    "IEEE 802.1X-2020 Clause 12.5.1";
}
leaf terminate-cause {
  type enumeration {
    enum common_port_MAC_operational_false {
      description
        "Common Port for this PAE is not operational.";
    }
    enum system_access_control_disabled {
      description
        "The system-access-control node of the pae-system
        is disabled or initialization process of this PAE
        is invoked.";
    }
    enum eapol_logoff_rx {
      description
        "The PAE has received EAPOL-Logoff frame.";
    }
    enum eap_reauthentication_failure {
      description
        "EAP reauthentication has failed.";
    }
    enum mka-failure_termination {
      description
        "MKA failure or other MKA termination.";
```

```
            }
            enum new_session-beginning {
              description
                "New session beginning.";
            }
            enum not_terminated_yet {
              description
                "Not Terminated Yet.";
            }
          }
          description
            "The reason for the session termination.";
          reference
            "IEEE 802.1X-2020 Clause 12.5.1";
        }
      }
    }
  }

  container nid-group {
    description
      "Contains both configuration and operational state nodes
      associated with the PAE NID group.";
    uses nid-group;
  }

}
```

14.6 YANG data model use in network access control applications

14.6.1 General

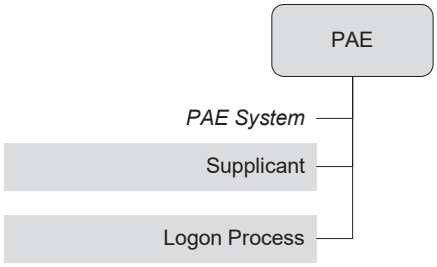
The YANG data modules specified in Clause 14 have been designed to support the full range of network access control applications described in Clause 7 of this standard and Clause 11 of IEEE Std 802.1AE-2018. Subclause 14.6 provides more detail on how these modules are used, in conjunction with the YANG data models designed to support IEEE Std 802.1Q, to support the following applications. The use cases in 14.6.2 to 14.6.6 are representative scenarios. However, the YANG data model covers all the use cases.

NOTE—It is also likely that Connectivity Fault Management (CFM) as specified in IEEE Std 802.1Q will be used to monitor the link.

14.6.2 Host access with a physically secure point-to-point LAN (7.1)

The PAE System YANG data nodes should be supported by the host. However, since the connection between the host and network access point is assumed to be physically secure, there is no need to support the KaY configuration nor operational YANG data nodes within the PAE. Additionally, since the PAE of the host is fulfilling the role of a supplicant, YANG configuration nodes associated with an authenticator are not required.

Figure 14-13 illustrates the key YANG data nodes required to support this application.



YANG nodes shaded in gray are containers which encompass additional configuration or operational attributes .
Items that are italicized are configuration or operational attributes .

Figure 14-13—IEEE 802.1X YANG model for host (7.1)

NOTE 1—This application does not require a MAC security entity in the interface stack. Instead there would be a PAC shim in the interface stack. The location of this shim in the interface stack is derived from the higher-layer-if and lower-layer-if attributes found in the IETF Interface Management YANG data model (Figure 14-4), which the PAE YANG data model augments.

The following is a specific YANG configurable attribute to support this application:

— If the host desires to utilize the logon process for managing the use of authentication credentials, then the YANG *logon-process:logon* configurable attribute needs to be *true*.

NOTE 2—The default settings of the YANG configurable attributes are intended to ease the operational expenditure to support basic applications. For example, by default, the *virtual-port-enable* attribute is *false*. In addition, the configuration attributes contained within the *Supplicant* node have defaults set to minimize the support of basic applications.

14.6.3 Network access point supporting a physically secure point-to-point LAN (7.1)

The PAE System YANG data nodes should be supported by the network access point. However, since the connection between the host and network access point is assumed to be physically secure, there is no need to support the KaY configuration or operational YANG data nodes within the PAE. Additionally, since the PAE of the network access point is fulfilling the role of an authenticator, YANG configuration nodes associated with a supplicant are not required.

Figure 14-14 illustrates the key YANG data nodes required to support this application.

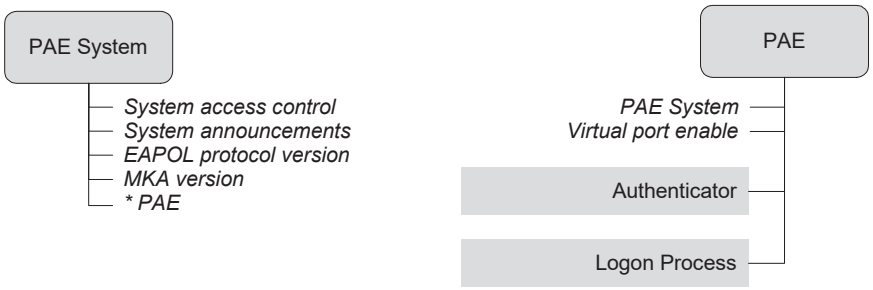


Figure 14-14—IEEE 802.1X YANG model for network access point (7.1)

NOTE 1— The default settings of the configurable attributes are intended to ease the amount of explicit configuration that is required. For example, by default, the *virtual-port-enable* attribute is *false*. In addition, the configuration attributes contained within the *Authenticator* node have defaults set to minimize the support basic applications.

NOTE 2— This application does not require a MAC security entity in the interface stack. Instead there would be a PAC shim in the interface stack. The location of this shim in the interface stack is derived from the higher-layer-if and lower-layer-if attributes found in the IETF Interface Management YANG data model (Figure 14-4), which the PAE YANG data model augments.

The following is a specific YANG configurable attribute to support this application.

— If the network access point desires to utilize the logon process for managing the use of authentication credentials, then the YANG *logon-process:logon* configurable attribute needs to be *true*.

14.6.4 Network access point supporting MACsec on a point-to-point LAN (7.3)

The PAE System YANG data nodes should be supported by the network access point. However, since the connection between the host and network access point is not secure, MACsec is needed. As a consequence, there is a need to support the KaY configuration and operational YANG data nodes within the PAE. Additionally, since the PAE of the network access point is fulfilling the role of an authenticator, YANG configuration nodes associated with a supplicant are not required.

Figure 14-15 illustrates the key YANG data nodes required to support this application.

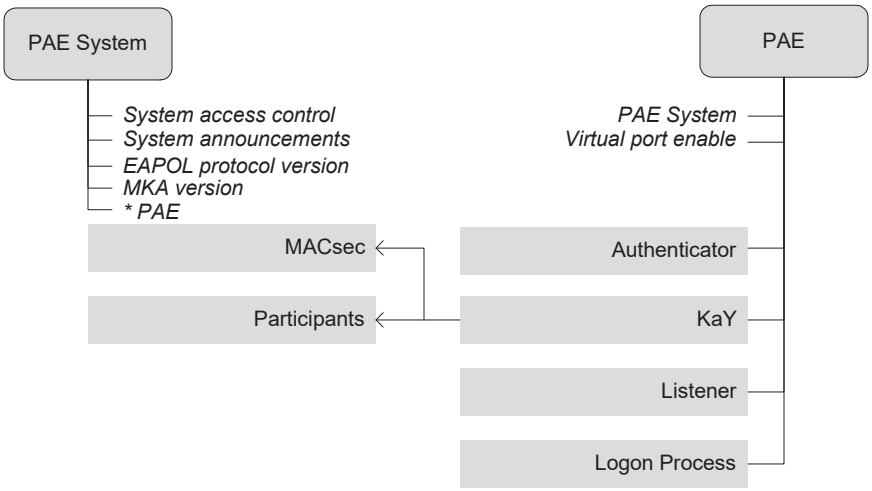


Figure 14-15—IEEE 802.1X YANG model for network access point (7.3)

NOTE 1—The default settings of the configurable attributes are intended to ease the amount of explicit configuration that is required. For example, by default, the *virtual-port-enable* attribute is *false*. In addition, the configuration attributes contained within the *Authenticator* and *KaY* nodes have defaults set to minimize the support basic applications.

NOTE 2—This application does require a MAC security entity in the interface stack. Consequently, there would be a SecY shim in the interface stack. The location of this shim in the interface stack is derived from the higher-layer-if and lower-layer-if attributes found in the IETF Interface Management YANG data model (Figure 14-4), which the PAE YANG data model augments.

The following are specific YANG configurable attributes to support this application:

— Enable the key agreement functionality (e.g., MACsec) by setting the YANG *KaY:enable* configurable attribute to *true*.

— Set the actor priority by setting the YANG *KaY:actor:priority* configurable attribute.

— Set the key server priority by setting the YANG *KaY:key-server:priority* configurable attribute.

NOTE 3—The default settings of the YANG configurable attributes of *KaY:group:join* is *true*, *KaY:group:form* is *false*, and *KaY:group:new* is *false*.

— Enable MACsec by setting the YANG *KaY:macsec:capable* configurable attribute to *true*.

NOTE 4—The default settings of the YANG configurable attributes of *KaY:suspend-on-request* is *true* and *KaY:suspend-for* is *0*.

NOTE 5—The default setting of the YANG configurable attribute *KaY:participants:retain* is *false*.

— If the network access point desires to receive EAPOL announcements (e.g., from a supplicant), then the YANG *listener:enable* configurable attribute needs to be *true*.

— If the network access point desires to utilize the logon process for managing the use of authentication credentials, then the YANG *logon-process:logon* configurable attribute needs to be *true*.

14.6.5 Network access point supporting MACsec on a multi-access LAN (7.5)

The key YANG data nodes required to support the application described in 7.4 and 7.5 are the same. The PAE System YANG data nodes should be supported by the network access point. However, since the connection between the host and network access point is not secure, MACsec is needed. As a consequence, there is a need to support the KaY configuration and operational YANG data nodes within the PAE. Refer to Figure 7-11. Additionally, since the PAE of the network access point is fulfilling the role of an authenticator, YANG configuration nodes associated with a supplicant are not required.

Figure 14-15 illustrates the key YANG data nodes required to support this application.

NOTE 1—This application does require a MAC security entity in the interface stack. Consequently, there would be a SecY shim in the interface stack.

The following are specific YANG configurable attributes to support this application:

— Enable the virtual port capabilities associated with the PAE by setting the YANG *pae:vp-enable* configurable attribute to *true*.

— Enable the key agreement functionality (e.g., MACsec) by setting the YANG *KaY:enable* configurable attribute to *true*.

— Set the actor priority by setting the YANG *KaY:actor:priority* configurable attribute.

— Set the key server priority by setting the YANG *KaY:key-server:priority* configurable attribute.

NOTE 2—The default settings of the YANG configurable attributes of *KaY:group:join* is *true*, *KaY:group:form* is *false*, and *KaY:group:new* is *false*.

— Enable MACsec by setting the YANG *KaY:macsec:capable* configurable attribute to *true*.

NOTE 3—The default settings of the YANG configurable attributes of *KaY:suspend-on-request* is *true*, and *KaY:suspend-for* is *0*.

NOTE 4—The default setting of the YANG configurable attribute of *KaY:participants:retain* is *false*.

— If the network access point desires to utilize the logon process for managing the use of authentication credentials, then the YANG *logon-process:logon* configurable attribute needs to be *true*.

14.6.6 Network access point supporting MACsec over LAG (11.5 of IEEE Std 802.1AE-2018)

The PAE System YANG data nodes should be supported by the network access point. In the host and in the network access point, an aggregation port provides an interface to each of their interconnecting links. Since those links are not secure, the PAEs for each of the connected aggregation ports mutually authenticate (and thus allow both host and network access point to confirm that those ports provide connectivity to the same peer system and are thus candidates for aggregation). The KaY associated with each PAE then uses MKA to initiate and maintain the MACsec connectivity supported by each aggregation port's SecY. PAE and KaY configuration and operational nodes thus need to be associated with the YANG interface that represents the aggregation port. The higher-layer-if attribute for each of those aggregation ports identifies the aggregator, which is the port (interface) that represents the connectivity provided by the LAG to its clients (to a bridge component's MAC Relay Entity, for example).

Figure 14-15 provides an illustration of the key YANG data nodes required to support this application.

The following are specific YANG configurable attributes to support this application:

— Enable the key agreement functionality (e.g., MACsec) by setting the YANG *KaY:enable* configurable attribute to *true*.

— Set the actor priority by setting the YANG *KaY:actor:priority* configurable attribute.

— Set the key server priority by setting the YANG *KaY:key-server:priority* configurable attribute.

NOTE 1—The default settings of the YANG configurable attributes of *KaY:group:join* is *true*, *KaY:group:form* is *false*, and *KaY:group:new* is *false*.

— Enable MACsec by setting the YANG *KaY:macsec:capable* configurable attribute to *true*.

NOTE 2—The default settings of the YANG configurable attributes of *KaY:suspend-on-request* is *true* and *KaY:suspend-for* is *0*.

NOTE 3—The default settings of the YANG configurable attributes of *KaY:participants:retain* is *false*.

— If the network access point desires to receive EAPOL announcements (e.g., from a supplicant), then the YANG *listener:enable* configurable attribute needs to be *true*.

— If the network access point desires to utilize the logon process for managing the use of authentication credentials, then the YANG *logon-process:logon* configurable attribute needs to be *true*.

Annex A

(normative)

PICS proforma[27]

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 implementor, 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

[27] *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 clauses, 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 clause 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 Conformation 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 situation described above 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 802.1X

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	
Information necessary to identify each real. port for which conformance is claimed	
NOTES 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification. 2—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 802.1X

Identification of protocol specification	IEEE Std 802.1X-2020, IEEE Standards for Local and Metropolitan Area Networks: Port-based Network Access Control
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	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 802.1X-2020.)	No [ ] Yes [ ]

Date of Statement	

A.5 Major capabilities and options

Item	Feature	Status	References	Support
pae	Are the mandatory functions for a PAE for each real port implemented?	M	5.3, 5.4, 12, A.6	Yes [ ]
supp	Is PAE functionality for an EAP/PACP Supplicant implemented?	O.1	5.3, 5.6, 12, 8, 11, A.7	Yes [ ] No [ ]
auth	Is the PAE functionality for an EAP/PACP Authenticator implemented?	O.1	5.3, 5.8, 12, 8, 11, A.8	Yes [ ] No [ ]
mka	Is the PAE functionality for MACsec Key Agreement implemented?	O.1	5.3, 5.10, 12, 9, A.9	Yes [ ] No [ ]
announce	Is the PAE capable of transmitting EAPOL announcements?	O	5.14, 12, 10, A.10	Yes [ ] No [ ]
paeListen	Is the PAE capable of listening to EAPOL announcements?	O	5.16, A.11	Yes [ ] No [ ]
mgt	Does the implementation support remote PAE management?	O	5.18, 13, A.12	Yes [ ] No [ ]
vp	Is the PAE capable of implementing virtual ports?	O	5.12, 12, A.13	Yes [ ] No [ ]
pac	Is a PAC implemented for each port that does not implement a SecY?	M	5.3, 5.18, 6.4, A.14	Yes [ ] N/A [ ]
yang	Can the implementation be configured or managed using YANG	O	14, A.15	Yes [ ] No [ ]

A.6 PAE requirements and options

Item	Feature	Status	References	Support
eapol	Are EAPOL PDUs encoded, decoded, addressed, and validated as specified?	M	11	Yes [ ]
paeRc	Is the Logon Process functionality implemented as specified in 12.5?	M	5.4, 12.5	Yes [ ]
paeRd	Is the CP state machine implemented as specified in 12.4.?	M	12.4	Yes [ ]
paeRe	Are the EAPOL frame reception statistics maintained and can they be retrieved?	M	5.4, 12.8.1	Yes [ ]
paeRf	Are the EAPOL frame reception diagnostics maintained and can they be retrieved?	M	5.4, 12.8.2	Yes [ ]
paeRg	Are the EAPOL frame transmission statistics maintained and can they be retrieved?	M	5.4, 12.8.3	Yes [ ]
paeRh	Are the system configuration functions specified in 12.9 supported?	M	5.4, 12.8.3	Yes [ ]
paeRi	Are a CAK and CKN derived from using EAP information as specified?	(supp OR auth) AND mka:M	5.4, 6.2.2	Yes [ ] N/A [ ]
paeRj	Specify the group address used:	M	5.4	_____
paeRk	Are EAPOL Packet Types other than Announcements transmitted using a MACsec protected Controlled Port?	X	5.4, 10.2	No[ ]

A.7 Supplicant requirements and options

Item	Feature	Status	References	Support
suppRa	Is EAP used as an authentication protocol, with PACP implemented as specified by the state machine, variables, and interfaces of Clause 8?	supp:M	5.6, 8.7	Yes [ ] N/A [ ]
suppRb	Are EAP methods that do not meet the requirements of 8.11 used?	X	5.6, 8.11	No [ ]
suppO1	Does the Supplicant support use of a Secure Device Identifier?	supp:O	5.7, 5.7.1	Yes [ ] N/A [ ]
suppO1a	Is the EAP method-EAP TLS implemented as specified?	supp01:M	5.7.1, 8.11.2	Yes [ ] N/A [ ]

A.8 Authenticator requirements and options

Item	Feature	Status	References	Support
authRa	Is EAP used as an authentication protocol, with PACP implemented as specified by the state machine, variables, and interfaces of Clause 8?	auth:M	5.6, 8.7	Yes [ ] N/A [ ]
authRb	Are EAP methods that do not meet the requirements of 8.11 used?	X	5.6, 8.11	No [ ]
authRc	Does the implementation support configuration of reAuthEnabled and reAuthPeriod?	auth:M	5.8, 8.6, 8.9	Yes [ ] N/A [ ]
authOa	Are the Authenticator diagnostic counters maintained and can they be retrieved?	auth:O	5.9, 8.10	Yes [ ] No [ ]
authO1	Does the Authenticator support use of a Secure Device Identifier?	auth:O	5.9, 5.9.1	Yes [ ] N/A [ ]
authO1a	Is the EAP method-EAP TLS implemented?	authO1:M	5.7.1, 8.11.2	Yes [ ] N/A [ ]

A.9 MKA requirements and options

Item	Feature	Status	References	Support
mkaV	Specify the MKA Version implemented.	mka:M	5.10	_____
mkaRa	Can the PAE maintain 2 or more simultaneous MKA instances?	mka:M	5.10, 9	Yes [ ] N/A [ ]
mkaRb	Specify the minimum number of simultaneous MKA instances supported:	mka:M	5.10, 9	_____
mkaRc	Can the PAE create, delete, and activate MKA participants as specified?	mka:M	5.10, 9.13, 9.16	Yes [ ] N/A [ ]
mkaRd	Can the PAE receive and use Group CAKs distributed by a Key Server?	mka:M	5.10	Yes [ ] N/A [ ]
mkaRe	Can the PAE encode parameters, and validate, encode, and decode MKPDUs as specified?	mka:M	5.10, 11.11	Yes [ ] N/A [ ]
mkaRf	Can the PAE use 128 bit CAKs and derived keys as specified?	mka:M	5.10, 6.2, 9.3	Yes [ ] N/A [ ]
mkaRg	Does the PAE follow the specified restrictions on use and disclosure of the CAK and derived keys?	mka:M	5.10, 6.2, 9.16	Yes [ ] N/A [ ]

A.9 MKA requirements and options (*continued*)

Item	Feature	Status	References	Support
mkaO1	Can the PAE use 256 bit CAKs and derived keys as specified?	mka:M	5.11, 6.2, 9.3	Yes [ ] N/A [ ]
mkaO2	Does the PAE support PSKs?	mka:O	5.11.1	Yes [ ] No [ ]
mkaO2a	Can PSKs be configured in the CAK cache?	mkaO2:M	6.3.3, 12.6	Yes [ ] No [ ]
mkaO3	Does the PAE support Group CAs as an MKA Key Server?	mka:O	5.11.2	Yes [ ] No [ ]
mkaO3a	Can the PAE operate as an MKA Key Server?	mkaO3:M	9.5	Yes [ ] N/A [ ]
mkaO3b	Can the PAE distribute a group CAK using an EAP derived pairwise CAK?	(mkaO3 AND auth):M	6.2, 9.12	Yes [ ] N/A [ ]
mkaO3c	Can the PAE distribute a group CAK using a PSK?	(mkaO2 AND mkaO3):M	6.2, 6.3.3, 9.12	Yes [ ] N/A [ ]
mkaO4	Does the PAE implement a CAK cache?	mka:O mkaO2:M	5.11.3	Yes [ ] No [ ]
mkaO4a	Is a lifetime associated with each cached CAK?	mkaO4:M	12.6	Yes [ ] N/A [ ]
mkaO4b	Are a CKN, KMD, and NID associated with each cached CAK?	mkaO4:M	12.6	Yes [ ] N/A [ ]
mkaO4c	Are group CAKs distributed by an MKA Key Server cached?	mkaO4:M	12.6	Yes [ ] N/A [ ]
mkaO4d1	Are pairwise CAKs derived from EAP exchanges cached?	(mkaO4:M AND (supp OR auth)):M	12.6	Yes [ ] N/A [ ]
mkaO4e	Can the PAE cache CAKs derived from EAP exchanges, even if prohibited by 12.6.	mkaO4:X	12.6	No [ ]
mkaO4f	Can the PAE distribute a KMD for a Group CAK not distributed by itself?	mkaO4:X	12.6	No [ ]
mkaO5	Does the PAE support in-service upgrades?	mka:O	5.11.4	Yes [ ] No [ ]
mkaO5a	Is the PAE capable of suspending MKA operation?	mkaO5:M	5.11.4, 9.18	Yes [ ] N/A [ ]
mkaO5b	Does the PAE communicate the most significant bits of each PN for XPN Cipher Suites?	mkaO5:M	5.11.4, 9.18.5	Yes [ ] N/A [ ]
mkaO5c	Does the PAE terminate an MKA suspension as specified when acting as Key Server?	mkaO5:M	9.18.4	Yes [ ] N/A [ ]
mkaO5d	Does the PAE include the specified parameters in each MKPDU transmitted?	mkaO5:M	9.18.5	Yes [ ] N/A [ ]
mkaO6	Does the PAE use additional protocol to coordinate in-service upgrades?	mkaO5:O	5.11.4, 9.18.6	Yes [ ] No [ ]
mkaO6a	Are directly set suspension parameter values consistent with MKA communication?	mkaO6:M	9.18.6	Yes [ ] N/A [ ]

A.10 Announcement transmission requirements

Item	Feature	Status	References	Support
announceRa	Are NIDs supported?	announce:M	5.14, 10	Yes [ ] N/A [ ]
announceRb	Can generic EAPOL-Announcements be transmitted using the Uncontrolled Port?	announce:M	5.14, 10.1, 10.2	Yes [ ] N/A [ ]
announceRc	Are EAPOL-Announcements encoded as specified?	announce:M	5.14, 10.1, 10.2	Yes [ ] N/A [ ]
announceRd	Are announcement transmissions rate limited?	announce:M	5.14, 11.12	Yes [ ] N/A [ ]
announceRe	Can the PAE transmit EAPOL-Announcements through an unsecured Controlled Port?	X	5.14, 10.2	No [ ]

A.11 Announcement reception requirements

Item	Feature	Status	References	Support
listenRa	Are received announcements interpreted as specified?	listen:M	5.16, 10, 11	Yes [ ] N/A [ ]
listenRb	Can received announcements be used to select a NID for authentication?	listen:M	5.16, 10, 12.5	Yes [ ] N/A [ ]
listenRc	Can the PAE solicit announcements?	listen:M	5.16, 11.13	Yes [ ] N/A [ ]
listenRd	Can the PAE use received announcements to select NIDs and authentication procedures?	listen:M	5.16, 11.3	Yes [ ] N/A [ ]
listenRe	Can the PAE use announcements to select a cached CAK for a NID?	(listen AND mka):M	5.16	N/A [ ]
listenRf	Can the PAE be configured to ignore received announcements?	M	5.16, 10	Yes [ ] N/A [ ]
listenRg	Can the PAE use announcements to select an appropriate EAP credentials?	(listen AND supp):M	5.16, 10	Yes [ ] N/A [ ]

A.12 Management and remote management

Item	Feature	Status	References	Support
mgtM1	Does the implementation support access to the SMIv2 MIB module specified in Clause 13 using SNMP version v3 or higher?	mgt:M	5.3, 13	Yes [ ] N/A [ ]

A.13 Virtual ports

Item	Feature	Status	References	Support
vpRa	Specify the number of virtual ports that can be supported.	vp:M	5.12	_____
vpRb	Is the PAE capable of operating 2 or more MKAinstances per port?	vp:M	5.12	Yes [ ]
vpRc	Is each virtual port supported by a SecY?	vp:M	5.12	Yes [ ]

A.13 Virtual ports (*continued*)

Item	Feature	Status	References	Support
vpRd	Are virtual ports created and managed as specified?	vp:M	5.12, 6.3.6, 9.14, 12.7	Yes [ ]
vpRe	Are virtual ports created with Supplicant functionality?	X	5.12	Yes [ ]
vpRf	Can the PAE support simultaneous EAP exchanges for each virtual port?	vp:M	5.12	Yes [ ]
vpRg	Is each virtual port that is a Bridge Port supported as specified?	vp:M	7.6	Yes [ ]

A.14 PAC

Item	Feature	Status	References	Support
pacRa	Does the PAC provide an Uncontrolled and Controlled Port over a real Common Port?	pac:M	6.4	Yes [ ]

A.15 YANG requirements and options

Item	Feature	Status	References	Support
yangRa	Can the YANG model specified in Clause 14 be accessible using a network configuration protocol (e.g., NETCONF)?	yang:M	5.23	Yes [ ] N/A [ ]
yangRb	Can PAEs within a PAE System be configured using the YANG model described in Clause 14?	yang:M	5.23	Yes [ ] N/A [ ]
yangRc	Can PAEs within a PAE System have operational state information retrieved using the YANG model described in Clause 14?	yang:M	5.23	Yes [ ] N/A [ ]

Annex B

(informative)

Bibliography

[B1] Fowler, M., "UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition," Pearson Education Inc., Boston, 2004, ISBN 0-321-19368-7.

[B2] IEEE Std 802.1AEbw™-2013, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security—Amendment 2: Extended Packet Numbering.[28,29]

[B3] IEEE Std 802.1X™-2001, IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control.

[B4] IEEE Std 802.1X™-2004, IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control.

[B5] IETF RFC 2246, The TLS Protocol Version 1.0, Dierks, T., and Allen, C., January 1999.[30]

[B6] IETF RFC 2865, Remote Authentication Dial In User Service (RADIUS), Rigney, C., Willens, S., Rubens, A., and Simpson, W., June 2000.

[B7] IETF RFC 2866, RADIUS accounting, Rigney, C., June 2000.

[B8] IETF RFC 2869, RADIUS Extensions, Rigney, C., Willats, W., and Calhoun, P., June 2000.

[B9] IETF RFC 3410, Introduction and Applicability Statements for Internet Standard Management Framework, Case J., R. Mundy, R., Partain, D. and Stewart B., December 2002.

[B10] IETF RFC 3414, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), Blumenthal U., and Wijnen B., December 2002.

[B11] IETF RFC 3575, IANA Considerations for RADIUS, Aboba, B., July 2003.

[B12] IETF RFC 3579, RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP), Aboba, B., and Calhoun, P., September 2003.

[B13] IETF RFC 3580, IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Guidelines, Congdon, P., Aboba, B., Smith, A., Zorn, G., and Roese, J., September 2003.

[B14] IETF RFC 3748, Extensible Authentication Protocol (EAP), Blunk, L., Vollbrecht, J., Aboba, B., Carlson, J., and Levkowetz, H., June 2004.

[B15] IETF RFC 4017, Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs, Stanley, D., Walker, J., and Aboba, B., March 2005.

[B16] IETF RFC 4072, Diameter Extensible Authentication Protocol (EAP) Application, Eronen, P., Hiller, T., and Zorn, G., August 2005.

[28] IEEE publications are available from The Institute of Electrical and Electronics Engineers (https://standards.ieee.org).
[29] The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.
[30] IETF RFCs are available from the Internet Engineering Task Force (https://www.ietf.org).

[B17] IETF RFC 4675, RADIUS Attributes for Virtual LAN and Priority Support, Congdon, P., Sanchez, M., and Aboba, B., September 2006.

[B18] IETF RFC 4849, RADIUS Filter Rule Attribute, Congdon, P., Sanchex, M., and Aboba, B., April 2007.

[B19] IETF RFC 4949, Internet Security Glossary, Version 2, Shirey, R., August 2007.

[B20] IETF RFC 5176, Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS), Chiba, M., Dommety, G., Eklund, M., Mitton, D., and Aboba, B., January 2008.

[B21] IETF RFC 6087, Guidelines for Authors and Reviewers of YANG Data Model Documents, Bierman, A., January 2011.

[B22] IETF RFC 6241, Network Configuration Protocol (NETCONF), Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A., editors, June 2011.

[B23] IETF RFC 6242, Using the NETCONF Protocol over Secure Shell (SSH), Wasserman, M., June 2011.

[B24] IETF RFC 6536, Network Configuration Protocol (NETCONF) Access Control Model, Bierman, A., and Bjorklund, M., March 2012.

[B25] IETF RFC 6733, Diameter Base Protocol, Fajardo, V., Loughney, J., Zorn, G., and Arkko, J., October 2012.

[B26] IETF RFC 6696, EAP Extensions for the EAP Re-authentication Protocol (ERP), Cao, Z., He, B., Shi, Y., Wu, Q., and Zorn, G., July 2012.

[B27] IETF RFC 7170, Tunnel Extensible Authentication Protocol (TEAP) Version 1, Zhou, H., Cam-Winget, N., Salowey, J., and Hanna, S., May 2014.

[B28] IETF RFC 7268, RADIUS Attributes for IEEE 802 Networks, Aboba, B., Malinen, J., Congdon, P., Salowey, J., and Jones, M., July 2014.

[B29] IETF RFC 8044, RADIUS Attributes for IEEE 802 Networks, DeKok, A., January 2017.

[B30] ISO/IEC 8824:1990, Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1) (Provisionally retained edition).[31]

[B31] ISO/IEC 8825:1990, Information technology—Open Systems Interconnection—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) (Provisionally retained edition).

[B32] ISO/IEC 14882:2003, Programming languages—C++.

[31] ISO and ISO/IEC documents are available from the International Organization of Standardization (http://www.iso.org) and from the International Electrotechnical Commission (http://www.iec.ch). These documents are also available from the American National Standards Institute (http://ansi.org).

[B33] NIST Federal Information Processing Standard 140-2, Security Requirements for Cryptographic Modules, 3 December 2002.[32]

[B34] OMG Unified Modeling Language (OMG UML), Version 2.5, March 2015.[33]

[32] NIST Special Publications are available from the National Institute of Standards and Technology (https://csrc.nist.gov/). FIPS 140-2 is available at https://www.nist.gov/cmvp.

[33] OMG documents are available from the Object Management Group (https://www.omg.org/).

Annex C

(normative)

State diagram notation

State diagrams are used to represent the operation of the protocol by a number of cooperating state machines each comprising a group of connected, mutually exclusive states. Only one state of each machine can be active at any given time. Each state is represented in the state diagram as a rectangular box, divided into two parts by a horizontal line. The upper part contains the state identifier, written in upper case letters. The lower part, the state block, contains procedures that are executed on entry to the state.

All permissible transitions between states are represented by arrows, the arrowhead denoting the direction of the possible transition. Labels attached to arrows denote the condition(s) that must be met in order for the transition to take place. All conditions are expressions that evaluate to TRUE or FALSE; if a condition evaluates to TRUE, then the condition is met. The label UCT denotes an unconditional transition (i.e., UCT always evaluates to TRUE). A transition that is global in nature (i.e., a transition that occurs from any of the possible states if the condition attached to the arrow is met) is denoted by an open arrow; i.e., no specific state is identified as the origin of the transition. When the condition associated with a global transition is met, it supersedes all other exit conditions including UCT. The special global condition BEGIN supersedes all other global conditions. Once BEGIN is asserted it remains asserted, and completion of each state block is followed by reentry to that state, until all state blocks have executed to the point that variable assignments and other consequences of their execution remain unchanged.

On entry to a state, the procedures defined for the state (if any) are executed exactly once, in the order that they appear on the page. Each action is deemed to be atomic; i.e., execution of a procedure completes before the next sequential procedure starts to execute. The procedures in only one state block execute at a time, even if the conditions for execution of state blocks in different state machines are satisfied, and all procedures in an executing state block complete execution before the transition to and execution of any other state block occurs, i.e., the execution of any state block appears to be atomic with respect to the execution of any other state block and the transition condition to that state from the previous state is TRUE when execution commences. The order of execution of state blocks in different state machines is undefined except as constrained by their transition conditions. A variable that is set to a particular value in a state block retains that value until a subsequent state block modifies the value.

On completion of all of the procedures within a state, all exit conditions for the state (including all conditions associated with global transitions) are evaluated continuously until one of the conditions is met. The label ELSE denotes a transition that occurs if none of the other conditions for transitions from the state are met (i.e., ELSE evaluates to TRUE if all other possible exit conditions from the state evaluate to FALSE). Where two or more exit conditions with the same level of precedence become TRUE simultaneously, the choice as to which exit condition causes the state transition to take place is arbitrary. A UCT has the lowest level of precedence.

Where it is necessary to split a state machine description across more than one diagram, a transition between two states that appear on different diagrams is represented by an exit arrow drawn with dashed lines, plus a reference to the diagram that contains the destination state. Similarly, dashed arrows and a dashed state box are used on the destination diagram to show the transition to the destination state. In a state machine that has been split in this way, any global transitions that can cause entry to states defined in one of the diagrams are deemed to be potential exit conditions for all of the states of the state machine, regardless of which diagram the state boxes appear in.

Should a conflict exist between the interpretation of a state diagram and the textual description associated with the state machine, the state diagram takes precedence. The interpretation of the special symbols and

operators used in the state diagrams is as defined in Table C-1; these symbols and operators are derived from the notation of the "C++" programming language, ISO/IEC 14882 [B32]. If a Boolean variable is described in this clause as being set it has or is assigned the value TRUE, if clear the value FALSE.

Table C-1—State machine symbols

Symbol	Interpretation
( )	Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes.
;	Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal English language conventions for reading text.
=	Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., a = b = X the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator.
!	Logical NOT operator.
&&	Logical AND operator.
||	Logical OR operator.
if...then...	Conditional action. If the Boolean expression following the **if** evaluates to TRUE, then the action following the **then** is executed.
{statement 1, ... statement N}	Compound statement. Braces are used to group statements that are executed together as if they were a single statement.
!=	Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.
==	Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right.
<	Less than. Evaluates to TRUE if the value of the expression to the left of the operator is less than the value of the expression to the right.
>	Greater than. Evaluates to TRUE if the value of the expression to the left of the operator is greater than the value of the expression to the right.
>=	Greater than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either greater than or equal to the value of the expression to the right.
+	Arithmetic addition operator.
–	Arithmetic subtraction operator.

Annex D

(informative)

IEEE 802.1X EAP and RADIUS usage guidelines

IETF RFC 3580 [B13] comprises the technical material previously included in IEEE Std 802.1X-2004 Annex D. This annex provides additional information including pointers to other guidelines, either established or under development at the time of the completion of this standard.

D.1 EAP Session-Id

IETF RFC 5247 describes the Session-Id used by this standard (IEEE Std 802.1X) to derive the CKN. The Session-Id uniquely identifies an EAP session between an EAP peer and EAP server, and is the concatenation of the EAP Type Code (uniquely identifying the EAP method) with a temporally unique Method-Id obtained from the method. The Method-Id is typically constructed from nonces or counters used within the EAP method exchange.

The EAP Session-Id for EAP-TLS is specified in IETF RFC 5216.

IETF RFC 5247 and IETF RFC 4072 [B16] define the RADIUS EAP-Key-Name Attribute (Type 102) used to convey the EAP Session-Id.

D.2 RADIUS Attributes for IEEE 802 Networks

IETF RFC 7268 [B28] updates RFC 3580 [B13] and provides a way to convey the NID, Access Status information (10.1), and other parameters defined by this standard (for use in Announcements, or as authorization parameters together with an Access Accept).

Annex E

(informative)

Support for ‘Wake-on-LAN’ protocols

Wake-on-LAN is a widely deployed power saving technology that allows a system that has been otherwise powered down to supply a small amount of power to a LAN or other network adaptor, sufficient for that adaptor to receive and recognize a ‘magic packet’—typically any received frame that contains the MAC address of the receiving station repeated eight times—and to ‘wake’ the system, causing it to run its normal operating system and protocols. A system that is ‘asleep’ is unlikely to be capable of operating EAPOL, MKA, or MACsec: if it is attached to an access controlled port, that port has to be configured so that it can circumvent the (absence of) security and transmit the ‘magic packet’ through its Uncontrolled Port.

NOTE—The question of whether the sleeping system receives the ‘magic packet’ through an uncontrolled or other port is moot: the sleeping system unlikely to be using a full protocol interface stack.

This standard makes recommendations (7.1.3) for the addressing and protocol identification of wake-on-LAN packets, and other packets explicitly designed for transmission to unauthenticated systems, so that a bridge or other access point does not have to replace the simple approach of enabling or denying access through a Controlled Port by a set of arbitrary security filters.

Annex F

(informative)

Unsecured multi-access LANs

A number of commercially available access points have implemented a form of authenticated port-based network access control for hosts attached via a shared media LAN without the use of MACsec. The notional security offered does not meet the requirements of this standard, or of IEEE Std 802.1AE, and is predicated on the assumption that none of the hosts will transmit frames using the source MAC address of another, although those source addresses are readily observable to all stations attached to the shared medium. In practice an attack of that form is not difficult, and the service offered is better described as a helpful way to assign each host to the appropriate VLAN (given source MAC addressed based ingress controls in a bridge based access point, and records of users maintained by a AAA/RADIUS server) than as a security service. Notwithstanding that characterization, recognizing that the service provided has both been useful and could benefit from a clear migration to the use of MACsec (so both security and VLAN configuration goals can be met), this annex explains how the provisions of this revision of IEEE Std 802.1X could be applied in such an unsecured shared media environment.

The application of port-based network access control to the unsecured multi-host scenario can be modeled by a bridging access point using either host access with a multi-access LANs (7.5) or group host access (7.6) as a starting point. Using the former, consider the bridging access point port depicted in Figure 7-13 and assume the following:

a) Each SecY is replaced with a filter that ensures that only allows its Controlled Port (connected to the MAC Relay Entity through the VLAN tagging function) to:
   1) transmit frames with a single destination address, the individual address of the associated host,
   2) receive frames with the individual address of the host as the source address;
b) A filter is also added to the unsecured real port (to the left in the figure), so that the frames forwarded through that port to and from the MAC Relay Entity are restricted to those with a group address in their destination address field;
c) The Spanning Tree Protocol Entity is only attached to the real port.

This modified figure then emphasizes some aspects of the unsecured multi-access configuration, including the creation of a pseudo port (approximating a secured virtual port) and the associated PAE context on receipt of an EAPOL frame with the individual source address of a host just as specified for a virtual port (see 12.7), and the possibility of using VLAN tagging so that unicast data to and from each attached host can travel on a different VLAN within the secured network. If the attached hosts are assigned to separate VLANs, group addressed frames could also be relayed through each of the pseudo ports instead of through the real port. The modified figure also serves to illustrate some of the challenges posed by the fact that the unsecured filters only approximate virtual ports, e.g., preventing duplicate or inappropriate reception of group addressed frames by the attached hosts. It also does not explain the fact that the attached hosts can communicate directly with each other.

Alternatively consider group host access (see Figure 7-14) with a single real port. The access point's interface stack can then be based on that shown to the right in Figure 7-4, similarly modified as follows:

d) The PAC is replaced by a filter that restricts Controlled Port reception by the MAC Relay Entity to frames with one of a known set of unicast addresses in its source address field, an authentication context and successful exchange having previously taken place with the PAE for each of those addresses, just as specified for virtual ports (see 12.7). The filter allows all group addressed frames to be transmitted, without restriction, but discards frames destined to unicast addresses not in the known set.

The pseudo ports or known unicast address filters described in these models could equally well be created or re-instantiated by the use of MKA, with a cached or pre-shared CAK, thus providing an incrementally beneficial path to full MACsec deployment.

Announcements, both generic and specific, can be used with the unsecured multi-access arrangement described in this annex, just as if each of the pseudo ports were a virtual port.

Annex G

(informative)

Test vectors

This standard specifies a KDF (6.2.1), and key and key name derivations using that KDF. This annex provides examples of correct outputs for specified inputs for 128-bit and 256-bit keys. Input and output parameters are octet strings, shown with spaces added for legibility. Each octet is represented as hexadecimal value with the most significant bits to the left. All conversions to or from integers use network byte order.

G.1 KDF

The KDF takes as input a Key, a Label, some Context, and the Length (in octets) of the KDF Output.

The following example uses a 128-bit Key and requests a 16-octet (128-bit) result:

Key	1ab9024f a04a03fe b9024fa0 4a03fe11
Label	48492054 48455245
Context	01020104
Length	0080
Output	b57a0b05 f43e9600 c3c4d15c 1e3c26e8

NOTE—The KDF adds a null (0x00) after ‘Label’ as part of the input to the PRF, as specified in 6.2.1, but that null is not part of the input to the KDF.

The following example uses a 256-bit Key and requests a 32-octet (256-bit) result:

Key	3946ec36 f59017f1 267e914a bed2dbf6
	633f52ae 7e20309d 3eefdda4 073adfad
Label	48492054 48455245
Context	01020104
Length	0100
Output	0efd01e5 b03a0951 a6df9bbf fe419016
	ee40fdbf c3335ebf 92ea0380 2214a307

G.2 CAK Key Derivation

The KDF is used to derive a pairwise CAK from an EAP MSK (6.2.2). The inputs are the initial octets of the MSK (Key), "IEEE8021 EAP CAK" (Label), two MAC addresses ordered with the lesser first (Context), and the CAK Length. In the following examples the MAC addresses are 00-D0-B7-1A-77-17 and 00-1B-63-93-FC-BC.

For a 128-bit CAK (Output), the first 16 octets of the MSK (Key) are used.

```
Key         e68a1ab9  0313024f  da7a04a0  3fea010f
Label       49454545  38303231  20454150  2043414b
Context     001b6393  fcbc00d0  b71a7717
Length      0080

Output      135bd758  b0ee5c11  c55ff6ab  19fdb199
```

For a 256-bit CAK (Output), the first 32 octets of the MSK (Key) are used.

```
Key         3946ec36  f59017f1  267e914a  bed2dbf6
            633f52ae  7e20309d  3eefdda4  073adfad
Label       49454545  38303231  20454150  2043414b
Context     001b6393  fcbc00d0  b71a7717
Length      0100

Output      a29efdb6  3d6fba73  c65daab2  295340a8
            37a8886e  94a905b5  c9c7ef1d  9dbb297e
```

G.3 CKN Derivation

The KDF is used to derive a CKN (6.2.2) for each pairwise CAK derived from an EAP MSK.

The inputs are the initial octets of the MSK (Key), "IEEE8021 EAP CKN" (Label), the EAP Session ID followed by two MAC addresses ordered with the lesser first (Context), and the CKN Length (16 octets). In the following examples, the Session-ID is an EAP-TLS Session-ID (0x0d followed by two 32-byte random numbers as prescribed in RFC 5216, and the MAC addresses are 00-D0-B7-1A-77-17 and 00-1B-63-93-FC-BC. The CKN is the Output of the KDF.

For a 128-bit CAK, the first 16 octets of the MSK (Key) are used as the Key.

```
Key         e68a1ab9  0313024f  da7a04a0  3fea010f
Label       49454545  38303231  20454150  20434b4e
Session-ID  0d
            d075693f  54b2b2eb  01da61f0  af5d429b
            65b1ebca  f536fba3  50777598  571728f6
            30c5c8da  5475489a  d47b6e34  89a97372
            e5a8fd55  0617972c  020d42a3  b13a4eae
MAC1        001b6393  fcbc
MAC2        00d0b71a  7717
Length      0080

Output      96437a93 ccf10d9d fe347846 cce52c7d
```

For a 256-bit CAK, the first 32 octets of the MSK (Key) are used as the Key.

Key	3946ec36	f59017f1	267e914a	bed2dbf6
	633f52ae	7e20309d	3eefdda4	073adfad
Label	49454545	38303231	20454150	20434b4e
Session-ID	0d			
	d075693f	54b2b2eb	01da61f0	af5d429b
	65b1ebca	f536fba3	50777598	571728f6
	30c5c8da	5475489a	d47b6e34	89a97372
	e5a8fd55	0617972c	020d42a3	b13a4eae
MAC1	001b6393	fcbc		
MAC2	00d0b71a	7717		
Length	0080			
Output	7888f5d4	8ba8b24e	96bb95bd	8c7304ec

G.4 KEK Derivation

The KDF is used to derive a KEK from a CAK (9.3.3). The inputs are the CAK (Key), “IEEE8021 KEK” (Label), the first 16 octets of the CKN for the CAK (Context, Keyid), and the KEK Length. The following examples show the KEKs (Output) by the KDF using the 128-bit and 256-bit CAKs and the CKNs derived in the examples above (G.2, G.3).

Key	135bd758	b0ee5c11	c55ff6ab	19fdb199
Label	49454545	38303231	204b454b	
Context	96437a93	ccf10d9d	fe347846	cce52c7d
Length	0080			
Output	8f5a384c	15d6ae93	02b462e3	63d03ca6

Key	a29efdb6	3d6fba73	c65daab2	295340a8
	37a8886e	94a905b5	c9c7ef1d	9dbb297e
Label	49454545	38303231	204b454b	
Context	7888f5d4	8ba8b24e	96bb95bd	8c7304ec
Length	0100			
Output	71340e45	4c84a123	2aa7977d	5ed86f78
	f250f3f9	d53584b9	337ff0c6	dfdc9f96

G.5 ICK Derivation

The KDF is used to derive an ICK from a CAK (9.3.3). The inputs are the CAK (Key), “IEEE8021 ICK” (Label), the first 16 octets of the CKN for the CAK (Context), and the KEK Length. The following examples show the KEKs (Output) derived by the KDF using the 128-bit and 256-bit CAKs and the CKNs derived in the examples above (G.2, G.3).

Key	135bd758	b0ee5c11	c55ff6ab	19fdb199
Label	49454545	38303231	2049434b	
Context	96437a93	ccf10d9d	fe347846	cce52c7d
Length	0080			
Output	8f1c5cb1	c8ed2e5f	047906e0	473aad4d

Key	a29efdb6 3d6fba73 c65daab2 295340a8
	37a8886e 94a905b5 c9c7ef1d 9dbb297e
Label	49454545 38303231 2049434b
Context	7888f5d4 8ba8b24e 96bb95bd 8c7304ec
Length	0100
Output	98b8544d 7390a41e 50ef72e2 5b4a0365
	23c919e8 12918871 949b4812 3eab526e

G.6 SAK Derivation

The KDF may be used to derive SAKs using a CAK and other randomly generated values (9.8.1). The inputs are the CAK (Key), “IEEE8021 SAK” (Label), a Key Server generated nonce followed by the MKA MI-value list and the Key Number (KN) (Context), and the SAK Length. The following examples show SAKs (Output) derived by the KDF using the 128-bit and 256-bit CAKs derived in the examples above (G.2).

Key	135bd758 b0ee5c11 c55ff6ab 19fdb199
Label	49454545 38303231 2053414b
KS-nonce	01020304 05060708 090a0b0c 0d0e0f10
MI-value-list	cd421cf8 6ba45793 8657675b 01020304
	05060708 0d1f36cf
KN	00000001
Length	0080
Output	04520592 5831ae59 c14550ed 59cc003d

Key	a29efdb6 3d6fba73 c65daab2 295340a8
	37a8886e 94a905b5 c9c7ef1d 9dbb297e
Label	49454545 38303231 2053414b
KS-nonce	01020304 05060708 090a0b0c 0d0e0f10
	11121314 15161718 191a1b1c 1d1e1f00
MI-value-list	cd421cf8 6ba45793 8657675b 01020304
	05060708 0d1f36cf
KN	00000001
Length	0100
Output	bb692568 b287484a 5f3f4793 b0973227
	0d13dd81 8373c15b 3f2793fd c1948a37

IEEE SA
STANDARDS ASSOCIATION

RAISING THE WORLD'S STANDARDS

Connect with us on:

Twitter: twitter.com/ieeesa
Facebook: facebook.com/ieeesa
LinkedIn: linkedin.com/groups/1791118
Beyond Standards blog: beyondstandards.ieee.org
YouTube: youtube.com/ieeesa

standards.ieee.org
Phone: +1 732 981 0060

IEEE