SSCI assignment for MACsec XPN Cipher Suites

Mick Seaman

Background for an IEEE 802.1X Maintenance Request and a proposed response/change. Updated following review of the prior version and discussion at the July 2016 meeting.

1. Background

MACsec uses 96-bit IVs with its GCM based Cipher Suites[1], with each frame being protected by an IV that is unique for the secure association key (SAK) in use[2]. The Default Cipher Suite uses IVs that comprise a 64 bit SCI (Secure Channel Identifier, derived from a 48 bit MAC Address) and a 32 bit incrementing PN (packet number). XPN Cipher Suites use IVs that comprise a 32 bit SSCI (Short Secure Channel Identifier) and a 64 bit extended packet number, significantly reducing any requirement to use fresh SAKs as a consequence of PN exhaustion[3].

MKA (MACsec Key Agreement) ensures that each of the SCIs used with any given SAK is unique. Each SAK is distributed, by a Key Server, in an MKPDU that also contains the Key Server's Live List—a list of the other members of the CA (Secure Connectivity Association) that have proved their liveness (current possession of the CAK[4])—and only the Key Server and those listed members can transmit frames using that SAK. Any SAK distributed after a change in the Key Server's Live List changes has to be a fresh SAK.

The Live List comprises the 96-bit random MIs (Member Identifiers) each chosen by an MKA participant whenever it restarts or otherwise loses its historical memory. Thus a rebooted would be CA member cannot accidentally reuse an IV (restarting its packet number sequence as if using the SAK for the first time) with a given SAK, even if it receives a copy/duplicate of the Key Server's prior MKPDU[5].

Each CA member has an SCI and includes that in transmitted MKPDUs. So when a non-XPN Cipher Suite is used, each participant in the Key Server's Live List can construct the IVs it uses for transmission with that SCI and its own incrementing PN once it has received the MKPDU with the SAK. It will delay using the new SAK until the Key Server starts using it itself (the Key Server allows time for each member to install the new SAK) but does not have to wait until its own Live List matches that of the Key Server.

The SCIs of the Key Server and the other members in the Live List in the MKPDU used to distribute an SAK determine the SSCIs used with that SAK—the member with the numerically greatest SCI using SSCI 0x00000001 to transmit, that with the next greatest 0x00000002, and so on. So when an XPN Cipher Suite is used, each member also needs to know its place in that ordering before it can determine its own SSCI and start transmitting using that SAK[6]. It also needs to know the SSCIs of each of the other members so it can begin receiving from them. Once it has received live[7] MKPDUs from each of the other members it can perform the necessary calculations.

The SSCI determination described above did not require any changes to the MKA protocol, so the development of IEEE 802.1AEbw-2013 (the MAC Security amendment introducing XPN Cipher Suites) did not have to be coordinated with an 802.1X amendment. However 802.1X should be updated to match 802.1AE's use of SCIs to determine SSCIs, guarding against the possibility of any further changes to 802.1X. The update should also address any related considerations.

2. Performance and robustness

SSCI determination is never a problem for the Key Server: it must have received an MKPDU (with the transmitter's SCI) from each of the members in its own Live List. It is also never a problem for the other member of a point-to-point (two member) CA—it knows its own SCI[8] and receives the Key Server's SCI in the MKPDU distributing the SAK. However, if

[1]As recommended by standards and related documents referencing GCM documents.

[2]The security of symmetric cryptography is critically dependent on not reusing any given IV with the same key.

[3]In many deployments a policy of periodic reauthentication will result in a fresh CAK, prompting distribution of a fresh SAK before exhaustion can occur. However, general purpose implementations should be capable of rolling connectivity seamlessly from one SAK to its successor whether or not a fresh CAK is being used, as other deployment scenarios may use long-lived PSK (pre-shared key) CAKs.

[4]Connectivity Association Key. Often derived from an EAP MSK, but can also be provisioned as a pre-shared key (PSK).

[5]Its newly chosen MI will not match any of those in the Live List of the old MKPDU.

[6]Otherwise two different members might pick the same SSCI and hence use the same sequence of IVs with the SAK.

[7]Each MKPDU contains lists of the other members' MI and a recent value of an associated sequence number (MN), thus proving liveness to each of them.

[8]Or SCIs if it is using mutiple transmit SCs (see P802.1AEcg).

there are three or more members in a CA operating over partial or intermittent connectivity it is possible that some of them may be unable to transmit (being unable to determine their own transmit SSCI) or receive (lacking knowledge of all the SCIs they are unable to determine the ordering and hence which SSCI to pair with which SCI).

Of course using a (virtual) shared media LAN with flaky connectivity is not a recipe for success, but denying all connectivity between two CA members because of connectivity issues with a third is undesirable. Rather knowledge of the connectivity that does exist should be available to management to allow problems to be diagnosed. We are currently missing an opportunity here (see below) and should consider including a small amount of additional information in the management UML, the YANG model, and (possibly) in the MIB[9].

In the absence of any clarification in the 802.1X standard, there is a clear risk that non-interoperable implementation specific solutions will be developed, some of which might introduce new security exposures. Fortunately there is a clear (once explained) solution that does not change the SSCI determination rules specified in IEEE Std 802.1AEbw. This is to specify that:

a) The Live List in any MKPDU used to distribute an SAK be ordered, with the MI for the CA member with the numerically greatest SCI first, that for next greatest SCI next, and so on.
b) The parameter set used to encode that Live List also includes the SSCI to be used, by the transmitting Key Server, with that SAK.

The SSCI (for that SAK) used for transmission by a CA member (other than the Key Server) represented by a participant with an MI in that Live List is then:

- Equal to its position in the MKPDU’s Live List (i.e. 0x00000001 if it is first in the list, 0x00000002 if it is second, and so on) if that is numerically less than that of the Key Server SSCI (as conveyed in the Live List parameter set of the received MKPDU).
- Equal to its position in the MKPDU’s Live List plus 1, if that position is numerically equal to or greater than that of the Key Server’s SSCI.

Only one other technical change is required, bumping the MKA version number from 2 to 3 to indicate that the Live Peer List is ordered and encodes the Key Server’s SSCI. This change meets the requirements of the protocol version handling rules specified in 802.1X-2010 11.11[10]. The Live Peer List is still perfectly intelligible to, and needs to be present with an unchanged parameter set type for, implementations that only understand prior versions. No new parameter sets are needed.

3. Proposed text

The changes to 802.1X that should be included in 802.1Xck are attached to this note as suggested changes to Clauses 3, 5, 9, and 11, and to Annex A (the PICS). The text for Clauses 3 and 5 just show additions, that for the other clauses and the annex are intended to be complete replacement clauses—with the changes introduced for this maintenance request shown change barred—in the hope that will make it easiest for the editor.

[9]The PAE MIB has been updated by P802.1Xck to take account of other maintenance items. This is probably the last time the MIB will be revised, so its now or never for additional updates.
[10]See also 802.1Xbx-2014, which updated notes in this and referenced clauses.

3. Definitions

WARNING! This is not a complete replacement Clause 3[7].

Change the NOTE following the definition of Short Secure Channel Identifier (SSCI), as follows:

NOTE—IEEE Std 802.1AEbw-2013 specifie~~s~~d the calculation of SSCI and Salt values used by the IEEE Std 802.1AE GCM-AES-XPN Cipher Suites from other MKA values. The IEEE Std 802.1Xck amendment to this standard constrained the order of entries in the MKPDU Live Peer List to facilitate that calculation ().

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

5. Conformance

WARNING! This is not a complete replacement Clause 5. As compared to P802.1Xck it has (I believe) just included changes for clause 5.4 (which was not in D0.61.

5.10 MKA requirements

Change the initial text of 5.10 as follows:

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 11-7, 11.11.3).

9. MACsec Key Agreement protocol (MKA)

Change 9.2.2 as follows:

9.2.2 SC identification

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

Change 9.4.5 as follows:

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~~D.4~~), SecYs are placed. IEEE Std 802.1AE-2006 Clause 11 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 ~~one of the~~ a group address~~es specified in Table 11-1 and shall not be an individual 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~~, as specified above,~~ and the inclusion of destination and source MAC addresses within the ICV calculation, makes it more difficult to attack MKA from a distance.

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

9.8 SAK generation, distribution, and selection

Change the fifth paragraph of 9.8 and the NOTE following as follows:

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 Peers List.

NOTE 3—Inclusion in the Potential Peers 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.

9.10 SAK installation and use

Change the first two paragraphs of 9.10 as follows:

Each CA member's KaY uses the LMI supported by the SecY (IEEE Std 802.1AE-2006, 10.7) 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 the IEEE Std 802.1AEcg-2016 amendment to 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 ~~CA member~~ 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 ~~CA members~~ 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).

Insert the following new text immediately prior to 9.10.1, renumbering the prior NOTE as NOTE-2:

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.

NOTE 3—The XPN Cipher Suites and the derivation of SSCIs from SCI values were first specified in the IEEE Std 802.1AEbw-2013 amendment to IEEE Std 802.1AE.

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 list, so its own SSCI (corresponding to the position that it would otherwise occupy in the list) is also encoded in each MKPDU used to distribute an SAK for an XPN Cipher Suite. 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 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 value explicitly encoded in the MKPDU. The numerically lower SSCIs are assigned in order to each of the SAs identified by their transmitting participant's position in the MKPDU Live Peer List while the SAs associated with numerically higher SSCI follow also in order. So, for example, if the Key Server's transmit SA's SSCI is 0x00000002 and there are three participant's 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.

9.10.1 MKPDU application data

Insert the following new text after the existing text of 9.10.1:

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.

Change 9.11 as follows:

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 802.1~~D~~AC, 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.

11. EAPOL PDUs

11.1 EAPOL PDU transmission, addressing, and protocol identification

Change the introductory text of 11.1 and the accompanying NOTEs as follows:

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~~a-2003~~ ~~[B2]~~.

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

Change the second paragraph of 11.1.1, the following bullets, and the NOTE as follows:

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, and a given system could choose to use EAPOL in connectivity associations with potentially different scopes. Subclause 7.7 and IEEE Std 802.1AE-2006 Clause 11 discuss the use of MACsec to secure both access to a provider network and transmission between systems attached to that network. IEEE ~~Std 802.1D,~~ IEEE Std 802.1Q~~, and their amendments~~ recognize<u>s</u> 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~~ (IEEE Std 802.1D)~~, 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.

~~NOTE—TPMRs (Two Port MAC Relays) are being specified by IEEE Std 802.1aj [B3].~~

Change the last paragraph of 11.1.1, and Table 11-1 as follows:

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. IEEE Std 802.1AE specifies additional group addresses for use by PAEs associated with the Provider Edge Ports of Ethernet Data Encryption devices. <<The foregoing change is a placeholder, we will probably want to update this table as a maintenance item to include the EDE PAE group addresses once those have been allocated for P802.1AEcg.>>

Table 11-1—EAPOL group address assignments

Address assignment	Address value	Filtered by:		
		MAC Bridges & VLAN Bridges		
			Provider Bridges & Provider Backbone Bridges	
				TPMRs
Bridge group address	01-80-C2-00-00-00	Y		
PAE group address	01-80-C2-00-00-03	Y	Y	
Link Layer Discovery Protocol multicast address	01-80-C2-00-00-0E	Y	Y	Y

These addresses are assigned in ~~Table 7-9 of IEEE Std 802.1D and~~ Table 8-1, Table 8-2, and Table 8-3 of IEEE Std 802.1Q.
Their values are repeated here for information.

Change the last paragraph of 11.1.1, and Table 11-1 as follows:

11.11 EAPOL-MKA

Change the third paragraphs of 11.11 as follows:

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 ~~2~~ 3.

Insert a new NOTE after the existing NOTE-2 as follows:

NOTE 3—The MKA Version Identifier was incremented to 3 by the IEEE Std 802.1Xck-2017 amendment to this standard which did not add any new parameter sets but imposed an ordering on entries in the Live Peer List and added the Key Server SSCI to the Live Peer List parameter set (9.10, <Figure 11-9>).

Change Table 11-7 as follows:

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 11-11, Figure 11-12	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	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.

Change Figure 11-9 as follows:

Bit:	8	7	6	5	4	3	2	1	Octet:
	Parameter set type = 1 or 2								1
	~~X~~ Key Server's SSCI[a]								2
	X	X	X	X	Parameter set body length				3
	Parameter set body length (cont)								4
	Member Identifier								5 – 16
	Message Number								17 – 20
	Member Identifier								[b]
	Message Number								

Figure 11-9—Live Peer List and Potential Peer List parameter sets

[a] The Key Server's transmit SA's SSCI is encoded in MKPDUs containing a Distributed SAK parameter set for use with an XPN Cipher Suite, otherwise 0 is encoded.

[b] Member Identifier, Message Number tuples are repeated to the end of the parameter set.

11.11.3 Encoding MKPDUs

Change the first sentence of 11.11.3 as follows:

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:

Change the text of bullet c) of 11.11.3 as follows:

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.

Annex A

(normative)

PICS Proforma

Change A.4.2 and the accompanying table as follows:

A.4.2 Protocol summary, IEEE Std 802.1X~~-2009~~

Identification of protocol specification	IEEE Std 802.1X-20~~09~~10, 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. IEEE Std 802.1Xbx-2014 Amd. IEEE Std 802.1Xck 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-2009.~~)	No [ ] Yes [ ]

Date of Statement	

Change table A.5 as follows:

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.15	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 [ ]

A.5 Major capabilities and options*(continued)*

Item	Feature	Status	References	Support
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		Yes [ ] No [ ]

<<The ‘yang’ entry above is just a place-holder for now. There is probably a much better way of expressing this sentiment.>>

Change table A.6 as follows:

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 [ ]
~~paeRb~~	~~Are group addressed EAPOL PDUs transmitted using one, and only one, of the specified addresses?~~	~~M~~	~~5.4, Table 11-1~~	~~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[ ]

Change Table A.9 as follows:

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 [ ]
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 [ ]

A.9 MKA requirements and options *(continued)*

Item	Feature	Status	References	Support
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 [ ]

Insert new table for A.15 YANG requirements and options, as follows:

A.15 YANG requirements and options

Item	Feature	Status	References	Support
yangRa	Say what it does.	yang:M		Yes [ ] No [ ]
yangRb		yang:M		Yes [ ] No [ ]
yangOa		yang:O		Yes [ ] N/A [ ]
yangOb		yang:O		Yes [ ] N/A [ ]
yangOc		yang:O		Yes [ ] N/A [ ]
yangOd		yang:O		Yes [ ] No [ ]
yangOe		yang:O		Yes [ ] N/A [ ]

<<There is (I hope) a YANGish way of saying what an implementation includes (other than discovering that by prodding it at run time), and the first requirement should be for the supplier of the implementation to supply that information.>>