P802.11w™/D2.~~2~~3

Draft 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:

Amendment 5 : Protected Management Frames

EDITORIAL NOTE - In this redline version of the draft, inserted text is shown as Inserted Text ***and deleted text is shown as*** ~~*Deleted Text*~~***.***

EDITORIAL NOTE - Editorial notes are distinguished like this. They are not part of the amendment and will be removed before it is published.

~~***EDITORIAL NOTE - the amendment number will be inserted by IEEE-SA editorial staff during preparation for publication.***~~

EDITORIAL NOTE- This revision of the amendment is based on the following (baseline) documents:

— *802.11-2007*

— *802.11k D8.0*

— *802.11r D7.0*

— *802.11n D2.~~05~~07*

— *802.11y D4.0*

The editing instructions contained in this amendment define how to merge the material contained herein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italic***. Four editing instructions are used: ***change***, ***delete***, ***insert,*** and ***replace***. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instructions. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editorial notes will not be carried over into future editions because the changes will be incorporated into the base standard.

EDITORIAL NOTE - The following table is not part of the amendment, and will be removed before the document is finalized.

Table 0—Change history

Draft version	Date	Contributions and motions applied
1.01	Jan. 17, 2007	Apply all editorial comment resolutions per 11-06-1729r11.
1.02	Mar. 8, 2007	Apply passed motions per 11-06-1759r4 and 11-07-100r3.
1.03	Mar 9, 2007	Include submissions 11-07-218r0, 11-07-2115,9r0, 11-07-221r0, 11-07-220r0, 11-07-243r3,11-07-390r0, 11-07-393r0 comments from group 8, 10 thru 18. Rev draft to 1.03
2.0	Mar 20, 2007	Renumber to Draft 2.0
2.1	July 3, 2007	Incorporate updates per May 2007 (Montreal) meeting to resolve comments: 212, 282, 68,569,213,80,214,285,103,104, 646, 244, 215,289, 711,17,37,246,291,581,647,38,648,218,430,293,582,649,715,717,294, 297,651,299,720,721,300,90,302,723,728,309,470,309,729,448,49,752,731,732,315,733,437,303,107,367,652,157,440,653,108,304,442,450,111, 320,657,326,659,280
2.2	August 13, 2007	Incorporate editorial changes based on 11-07-714r10 and submissions 11-07-2051r2, 11-07-2241r0 (2nd page only), 11-07-243r7, 11-07-2239r0, 11-07-2244r3, 11-07-2240r1, 11-07-2238r1
2.3	Sept. 7, 2007	Incorporate further updates per 11-07/714r11 Group 7 and Group 8

3. Definitions

Insert the following definitions:

3.72a Integrity GTK (IGTK): A random value, assigned by the broadcast/multicast source, which is used to protect broadcast/multicast medium access control (MAC) management protocol data units (MMPDUs) from that source.

3.125a Robust Management frame: A management frame that is eligible for protection by the Robust Management frame service.

4. Abbreviations and acronyms

Insert the following new abbreviations and acronyms in alphabetical order:

BIP	Broadcast/Multicast Integrity Protocol
IGTK	Integrity GTK
IPN	IGTK packet number
MMIE	Management MIC Information Element

5. General description

5.2 Components of the IEEE 802.11architecture

5.2.3 Distribution system (DS) concepts

5.2.3.2 RSNA

Insert at the end of the dashed item list in 5.2.3.2:

— Enhanced cryptographic encapsulation mechanisms for Robust Management frames

5.4 Overview of the services

5.4.2.4 Disassociation

Change the 3rd paragraph of 5.4.2.4 as follows:

The disassociation service may be invoked by either party to an association (non-AP STA or AP). Disassociation is a notification, not a request. Disassociation cannot be refused by either party to the association~~.~~ except when Management Frame Protection is enabled and the disassociation message integrity check fails.

5.4.3 Access control and data confidentiality services

5.4.3.2 Deauthentication

Change the 2nd paragraph of 5.4.3.2 as follows:

In an ESS, because authentication is a prerequisite for association, the act of deauthentication shall cause the station to be disassociated. The deauthentication service may be invoked by either authenticated party (non-AP STA or AP). Deauthentication is not a request; it is a notification. Deauthentication shall not be refused by either party. ~~When an AP sends a deauthentication notice to an associated STA, the association shall also be terminated~~. The association at the transmitting STA is terminated when the STA sends a deauthentication notice to an associated STA. Deauthentication and subsequent disassociation is not refused by the receiving STA except when Management Frame Protection is enabled and the message integrity check fails.

Change the 4th paragraph of 5.4.3.2 as follows:

In an RSNA, deauthentication also destroys any related PTKSA, group temporal key security association (GTKSA), station to station link master key security association (SMKSA), ~~and~~ station to station link transient key security association (STKSA), and integrity group temporal key security association (IGTKSA) that exist in the STA and closes the associated IEEE 802.1X Controlled Port. If pairwise master key (PMK) caching is not enabled, deauthentication also destroys the pairwise master key security association (PMKSA) from which the deleted PTKSA was derived.

5.4.3.3 Data confidentiality

Change the text of 5.4.3.3 as follows:

In a wired LAN, only those STAs physically connected to the wire can send or receive LAN traffic. With a wireless shared medium, there is no physical connection, and all STAs and certain other RF devices in or near the LAN may be able to send, receive, and/or interfere with the LAN traffic. Any IEEE 802.11-compliant STA can receive all like-PHY IEEE 802.11 traffic that is within range and can transmit to any other IEEE 802.11 STA within range. Thus, the connection of a single wireless link (without data confidentiality) to an existing wired LAN may seriously degrade the security level of the wired LAN.

To bring the security of the WLAN up to the level implicit in wired LAN design, IEEE Std 802.11 provides the ability to protect the contents of messages. This functionality is provided by the data confidentiality service. Data confidentiality is an SS.

IEEE Std 802.11 provides three cryptographic algorithms to protect data traffic: WEP, TKIP, and CCMP. WEP and TKIP are based on the ARC4[14] algorithm, and CCMP is based on the advanced encryption standard (AES). A means is provided for STAs to select the algorithm(s) to be used for a given association.

IEEE 802.11 provides one cryptographic algorithm, CCMP, to protect unicast Robust Management frames.

The default data confidentialitty state for all IEEE 802.11 STAs is “in the clear”. If the data cofidentiality service is not invoked, all ~~messages~~ frames shall be sent uprotected. If this policy is unacceptable to the sender, it shall not send data frames; and if the policy is unacceptable to the receiver, it shall discard any received data frames. Unprotected ~~data~~ frames received at a STA configured for mandatory data confidentiality, as well as protected ~~data~~ frames using a key not available at the receiving STA, are discarded without an indication to LLC (or without indication to distribution services in the case of “To DS” frames received at an AP). These frames are acknowledged on the WM [if received withou frame check sequence (FCS) error] to avoid wasting WM bandwidth on retries of frames that are being discarded.

5.4.3.4 Key management

Change the text of 5.4.3.4 as follows:

The enhanced data confidentiality, data authentication, and replay protection mechanisms require fresh cryptographic keys and corresponding security associations. The procedures defined in this standard provide fresh keys by means of protocols called the 4-Way Handshake and Group Key Handshake.

5.4.3.5 Data origin authenticity

Change the text of 5.4.3.5 as follows:

The data origin authenticity mechanism defines a means by which a STA that receives a data or Robust Management frame can determine which STA transmitted the MAC protocol data unit (MPDU) or MAC management protocol data unit (MMPDU). This feature is required in an RSNA to prevent one STA from masquerading as a different STA. ~~This mechanism is provided for STAs that use CCMP or TKIP.~~

Data origin authenticity is only applicable to unicast data frames, and unicast Robust Management frames. The protocols do not guarantee data origin authenticity for broadcast/multicast data frames or broadcast/multicast Robust Management frames, as this cannot be accomplished using symmetric keys and public key methods are too computationally expensive.

5.4.3.6 Replay detection

Change the text of 5.4.3.6 as follows:

The replay detection mechanism defines a means by which a STA that receives a data or Robust Management frame from another STA can detect whether the received ~~data~~ frame is an unauthorized retransmission.

This replay protection mechanism is provided for data frames for STAs that use CCMP or TKIP. The replay protection mechanism is also provided for Robust Management frames for STAs that use CCMP and BIP.

Insert the following sub clause after 5.4.3.6 as follows:

5.4.3.6a Robust Management frame protection

The Robust Management ~~Frames~~ frames are Action frames, Disassociate and Deauthenticate frames.

Management Frame Protection protocols apply to Robust Management frames after RSNA PTK establishment for protection of unicast frames is completed and after delivery of the IGTKs to protect broadcast/multicast frames.

5.8 IEEE 802.11 and IEEE 802.1X

5.8.2 Infrastructure functional model overview

5.8.2.1 AKM operations with AS

Change the second paragraph of 5.8.2.1 as follows:

A 4-way Handshake utilizing EAPOL-Key frames is initiated by the Authenticator to do the following:

— Confirm that a live peer holds the PMK.

— Confirm that the PMK is current.

— Derive a fresh pairwise transient key (PTK) from the PMK.

— Install the pairwise encryption and integrity keys into IEEE 802.11.

— Transport the group temporal key (GTK) and GTK sequence number from Authenticator to Supplicant and install the GTK and GTK sequence number in the STA and, if not already installed, in the AP.

— If Robust Management Frame Protection is enabled, transport the integrity GTK (IGTK), and the IGTK sequence number from Authenticator to the Supplicant and install these values in the STA and, if not already installed, in the AP.

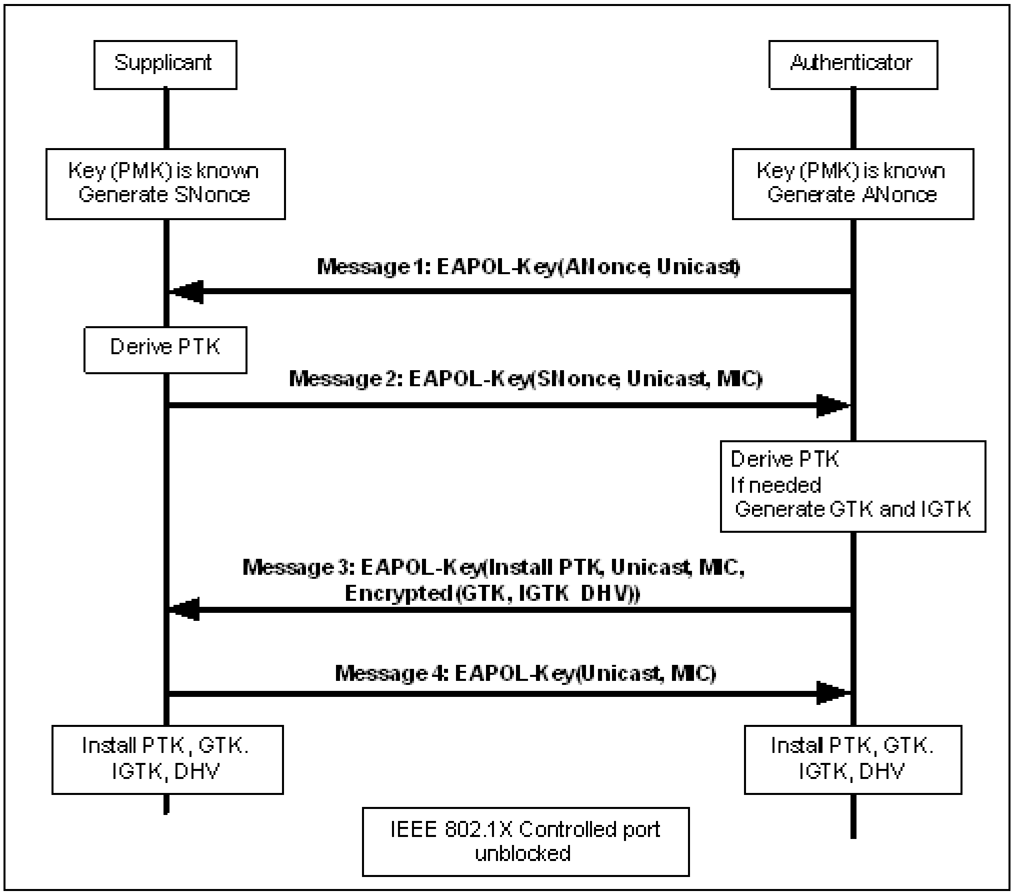
— Verify that the RSN capabilities negotiated are valid as defined in 7.3.2.25.3.

— Confirm the cipher suite selection.

Insert the following paragraph at the end of 5.8.2.1:

When Robust Management Frame Protection is enabled, the Authenticator also uses the Group Key Handshake with all associated STAs to change the IGTK. The Authenticator encrypts the GTK, and IGTK values in the EAPOL-Key frame as described in 8.5.

Replace Figure 5-13 with the following figure, with the changes being the inclusion of "IGTK" in message 3 and in both the Supplicant and Authenticator boxes that begin with "Install" and in the Authenticator box to "Generate GTK and IGTK":


Supplicant
Authenticator
Key (PMK) is known
Generate SNonce
Key (PMK) is known
Generate ANonce
Message 1: EAPOL-Key(ANonce, Unicast)
Derive PTK
Message 2: EAPOL-Key(SNonce, Unicast, MIC)
Derive PTK
If needed
Generate GTK and IGTK
Message 3: EAPOL-Key(Install PTK, Unicast, MIC,
Encrypted(GTK, IGTK DHV))
Message 4: EAPOL-Key(Unicast, MIC)
Install PTK, GTK.
IGTK, DHV
Instal PTK, GTK.
IGTK, DHV
IEEE 802.1X Controlled port
unblocked

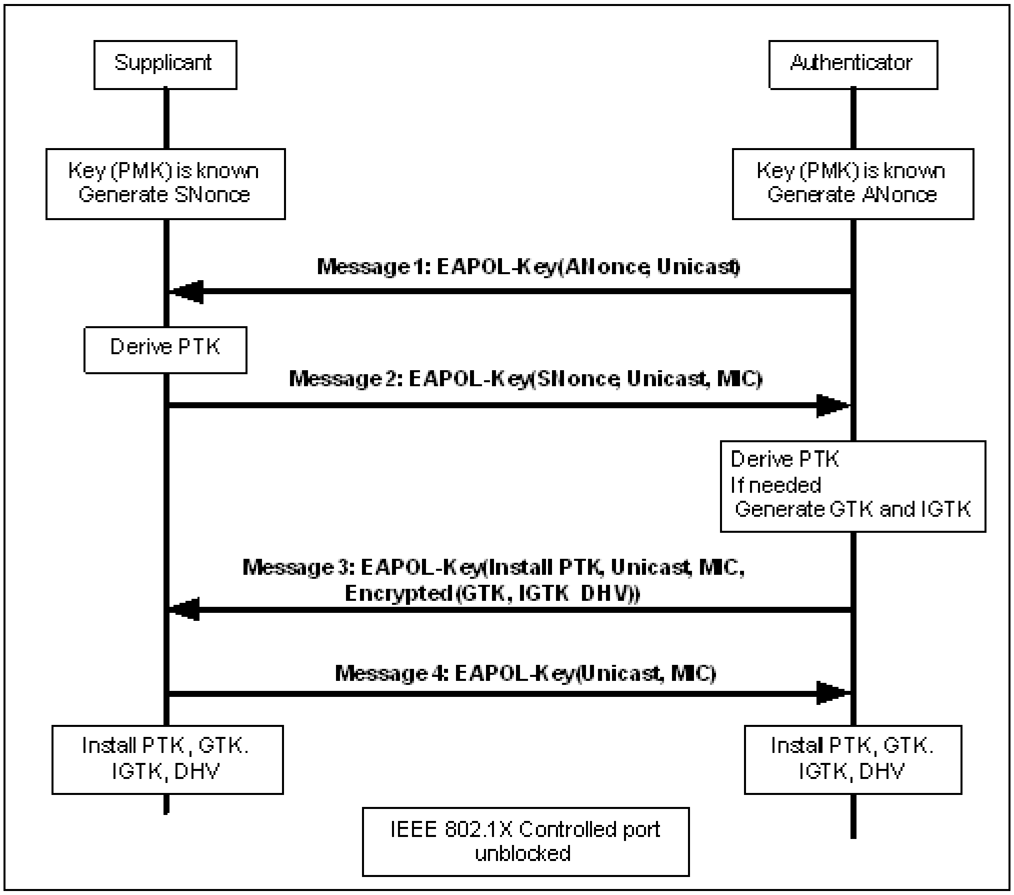
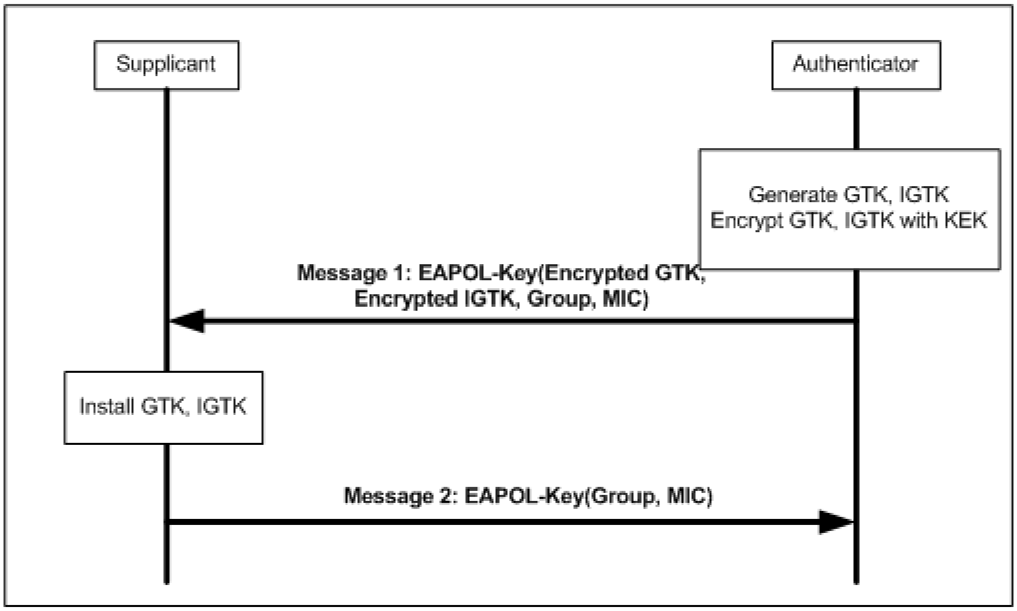

Supplicant
Authenticator
Key (PMK) is known
Generate SNonce
Key (PMK) is known
Generate ANonce
Message 1: EAPOL-Key(ANonce, Unicast)
Derive PTK
Message 2: EAPOL-Key(SNonce, Unicast, MIC)
Derive PTK
If needed
Generate GTK and IGTK
Message 3: EAPOL-Key(Install PTK, Unicast, MIC,
Encrypted(GTK, IGTK DHV))
Message 4: EAPOL-Key(Unicast, MIC)
Install PTK, GTK.
IGTK, DHV
Instal PTK, GTK.
IGTK, DHV
IEEE 802.1X Controlled port
unblocked

Figure 5-13—Establishing pairwise and group keys

Replace Figure 5-14 with the following figure, with the changes being the inclusion of "Encrypted IGTK" in Message 1, "IGTK" in the Authenticator box beginning with 'Generate" and "IGTK" in the Supplicant box beginning with "Install"; the update to the 2nd box on the right has intentionally fixed 802.11-2007 to correctly state "Encrypt GTK, IGTK with KEK":


Supplicant
Authenticator
Generate GTK, IGTK
Encrypt GTK, IGTK with KEK
Message 1: EAPOL-Key(Encrypted GTK, Encrypted IGTK, Group, MIC)
Install GTK, IGTK
Message 2: EAPOL-Key(Group, MIC)

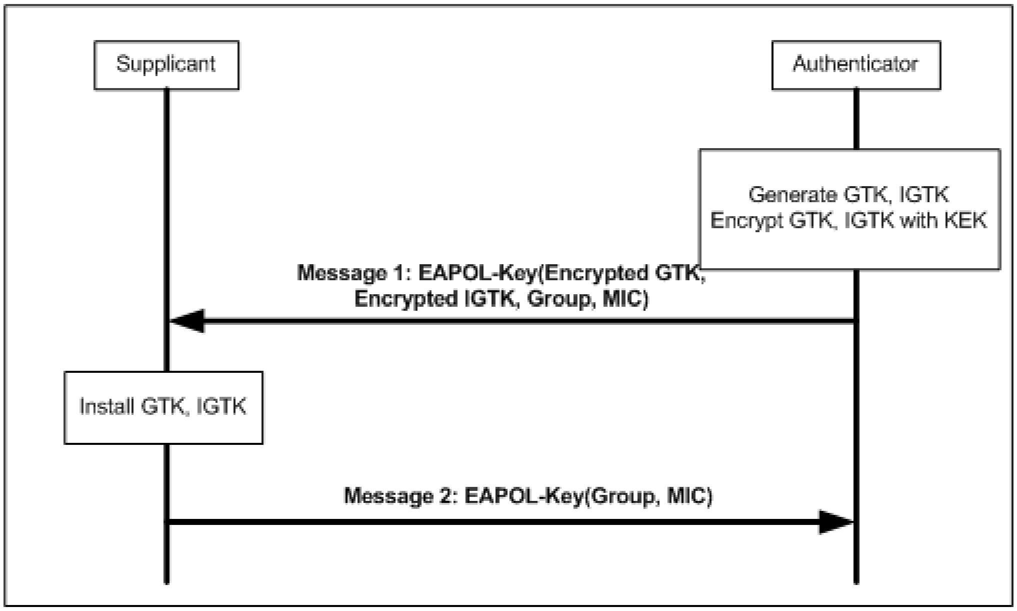

Supplicant
Authenticator
Generate GTK, IGTK
Encrypt GTK, IGTK with KEK
Message 1: EAPOL-Key(Encrypted GTK, Encrypted IGTK, Group, MIC)
Install GTK, IGTK
Message 2: EAPOL-Key(Group, MIC)

Figure 5-14 — Delivery of subsequent group keys

5.8.2.2 Operations with PSK

Insert a new item after the 3rd item in 5.8.2.2 as follows:

— If Robust Management Frame Protection is enabled, the IGTK and IGTK sequence number are sent from the Authenticator to the Supplicant just as in the AS case. See Figure 5-13— and Figure 5-14 —.

Insert a new sub clause after 5.8.5 as follows:

5.8.5a Protection of broadcast and multicast Robust Management frames

When Management Frame Protection is enabled, all broadcast/multicast Robust Management frames shall be encapsulated using the procedures defined in 11. This service provides integrity protection of broadcast/multicast Robust Management frames using BIP.

6. MAC service definition

6.1 Overview of MAC services

6.1.2 Security services

Change the text of 6.1.2 as follows:

Security services in IEEE Std 802.11 are provided by the authentication service and the TKIP, ~~and~~ CCMP and BIP mechanisms. The scope of the security services provided is limited to station-to-station data and Robust Management frame transmissions ~~exchange~~. The data confidentiality service offered by an IEEE 802.11 TKIP and CCMP implementation is the protection of the MSDU. When CCMP is used, the data confidentiality service is provided for the MPDU or unicast MMPDU. For the purposes of this standard, TKIP and CCMP are viewed as logical services located within the MAC sublayer as shown in the reference model, Figure 5-10 (in 5.7). Actual implementations of the TKIP and CCMP services are transparent to the LLC and other layers above the MAC sublayer.

The security services provided by TKIP and CCMP in IEEE Std 802.11 are as follows:

a) Data Confidentiality;

b) Authentication; and

c) Access control in conjunction with layer management~~;~~.

BIP provides authentication (integrity) and access control for broadcast/multicast Robust Management frames.

During the authentication exchange, both parties exchange authentication information as described in Clause 8 and 11A.

The MAC sublayer security services provided by TKIP, ~~and~~ CCMP and BIP rely on information from non-layer-2 management or system entities. Management entities communicate information to TKIP, ~~and~~ CCMP, and BIP through a set of MAC sublayer management entity (MLME) interfaces and MIB attributes; in particular, the decision tree for TKIP, ~~and~~ CCMP and BIP defined in 8.7 is driven by MIB attributes.

The use of WEP for confidentiality, authentication, or access control is deprecated. The WEP algorithm is unsuitable for the purposes of this standard.

The standard does not support the use of TKIP for Management Frame Protection.

7. Frame formats

7.1 MAC Frame formats

7.1.3 Frame fields

7.1.3.1 Frame control field

7.1.3.1.8 Protected frame field

Change the text of 7.1.3.1.8 as follows:

The Protected Frame field is 1 bit in length. The Protected Frame field is set to 1 if the Frame Body field contains information that has been processed by a cryptographic encapsulation algorithm. The Protected Frame field is set to 1 only within data frames, ~~and~~ within management frames of subtype Authentication and within unicast Robust Management frames. The Protected Frame field is set to 0 in all other frames. When the Protected Frame field is set to 1, the Frame Body field is protected utilizing the cryptographic encapsulation algorithm and expanded as defined in Clause 8. The Protected Frame field is set to 0 in Data frames of subtype Null Function, CF-ACK (no data), CF-Poll (no data), and CF-ACK+CF-Poll (no data) (see 8.3.2.2 and 8.3.3.1 that show that the frame body must be one octet or longer to apply the encapsulation).

7.2 Format of individual frame types

7.2.3 Management frames

7.2.3.3 Disassociation frame format

Change 7.2.3.3 including Table 7-9 (with the changes of Table 7-9 being the addition of the Management MIC IE) and a note at the end of the table as follows:

The frame body of a management frame of subtype Disassociation contains the information shown in Table 7-9.

Table 7-9—Disassociation frame body

Order	**Information**
1	Reason Code
2-(Last -1)	One or more vendor-specific information elements may appear in this frame.
Last	Management MIC IE (MMIE)

NOTE: The Management MIC IE appears after any fields that it protects. Therefore, it shold always appear last in the frame to protect contents of the entire frame.

7.2.3.11 Deauthentication

Change 7.2.3.11 including Table 7-18 (with the changes of Table 7-18 being the addition of the Management MIC IE as the "Last" value in the "Order" column and removing the 2nd sentence in the "Information" column of the new Order "2") row and a note at the end of the table as follows:

The frame body of a management frame of subtype Deauthentication contains the information shown in Table 7-18.

Table 7-18—Deauthentication frame body

Order	Information
1	Reason Code
2-(Last -1)	One or more vendor-specific information elements may appear in this frame. ~~This information element follows all other information elements.~~
Last	Management MIC IE (MMIE)

NOTE: The Management MIC IE appears after any fields that it protects. Therefore, it shold always appear last in the frame to protect contents of the entire frame.

7.2.3.12 Action frame format

Change 7.2.3.12 including Table 7-19 (with the changes of Table 7-19 being the addition of the Management MIC IE as the new order "Last", and removing the 2nd sentence of the resulting "Information" column in the new Order "2" row) and a note at the end of the table as follows:

The frame body of a management frame of subtype Deauthentication contains the information shown in Table 7-19.

Table 7-19—Action frame body

Order	Information
1	Action
2-(Last -1)	One or more vendor-specific information elements may appear in this frame. ~~This information element follows all other information elements.~~
Last	Management MIC IE (MMIE)

NOTE: The Management MIC IE appears after any fields that it protects. Therefore, it shold always appear last in the frame to protect contents of the entire frame.

7.3 Management frame body components

7.3.1 Fixed fields

7.3.1.7 Reason code field

Insert the following rows into Table 22 - Reason Codes before the “Reserved” entry and update the numbering appropriately:

Table 7-22—Reason Codes

Reason Code	Meaning
<ANA>	Invalid management group cipher
<ANA>	Robust management frame policy violation

EDITORIAL NOTE: The entry values are left as <ANA> for now, pending ANA assignment

7.3.2 Information elements

Change the last paragraph in 7.3.2 as follows:

A STA that encounters an unknown or reserved element ID value in a management frame received without error shall ignore that element and shall parse any remaining management frame body for additional information elements with recognizable element ID values. The frame body components specified for many management subtypes result in elements ordered by ascending element ID, with the exception of the MIC Management IE (7.3.2.54). The MIC Management IE must appear at the end of any Robust Management frame to protect the entire contents of the frame.

Insert the following row (ignoring the header row) in Table 26 - Element IDs in the correct position to preserve ordering by the “Element ID” column and update the “Reserved” range of codes appropriately:

Table 7-26—Element IDs

Information Element	Element ID	Length (in octets)
Management MIC (see 7.3.2.54 (MMIE))	<ANA>	18

EDITORIAL NOTE: <ANA> request to ANA for assignment of MMIE.

7.3.2.25 RSN information element

Change the first paragraph of 7.3.2.25 as follows:

The RSN information element contains authentication and pairwise cipher suite selectors, a single group data cipher suite selector, an RSN Capabilities field, the PMK identifier (PMKID) count, and PMKID list. If dot11RSNAProtectedManagementFramesEnabled is set to TRUE, a single management group cipher suite selector is optionally included. See Figure 89. All STAs implementing RSNA ~~shall~~ support this element. The size of the RSN information element is limited by the size of an information element, which is 255 octets. Therefore, the number of pairwise cipher suites, AKM suites, and PMKIDs is limited.

Replace Figure 7-72 with the following figure, where a new field **Group Management ~~Group~~ Cipher** ***is inserted at the end and "Data" inserted in the 4th column to read "Group Data Cipher Suite":***

Element ID	Length	Version	Data Group Cipher Suite	Pairwise Cipher Suite Count	Pairwise Cipher Suite List	AKM Suite Count	AKM Suite List	RSN Capabilities	PMKID Count	PMKID List	Group Management ~~Group~~ Cipher Suite
1	1	2	4	2	4*m	2	4*n	2	2	16*s	4

Figure 7-72—RSN Information Element format

Change the last paragraph of 7.3.2.25 as follows:

NOTE- The following represent sample information elements:

802.1X authentication, CCMP and data group cipher suites (WEP-40, WEP-104, and TKIP not allowed).:

30, // information element id, 48 expressed as Hex value
14, // length in octets, 20 expressed as Hex value
01 00, // Version 1
00 0F AC 04, // CCMP as data group key cipher suite
01 00, // pairwise key cipher suite count
00 0F AC 04, // CCMP as pairwise cipher suite
01 00, // authentication count
00 0F AC 01 // IEEE 802.1X authentication
00 00 // No capabilities

802.1X authentication, CCMP pairwise and group cipher suites (WEP-40, WEP-104 and TKIP not allowed), preauthentication supported:

30, // information element id, 48 expressed as Hex value
14, // length in octets, 20 expressed as Hex value
01 00, // Version 1
00 0F AC 04, // CCMP as data group key cipher suite
01 00, // pairwise key cipher suite count
00 0F AC 04, // CCMP as pairwise cipher suite
01 00, // authentication count
00 0F AC 01 // IEEE 802.1X authentication

01 00 // Preauthentication capabilities

802.1X authentication, Use GTK for pairwise cipher suite, WEP-40 group cipher suites, optional RSN Capabilities omitted:

30, // information element id, 48 expressed as Hex value

12, // length in octets, 18expressed as Hex value

01 00, // Version 1

00 0F AC 01, // WEP-40 as data group key cipher suite

01 00, // pairwise key cipher suite count

00 0F AC 00, // Use group key as pairwise cipher suite

01 00, // authentication count

00 0F AC 01 // IEEE 802.1X authentication

802.1X authentication, Use CCMP for pairwise cipher suite, CCMP group cipher suites, preauthentication and a PMKID:

30, // information element id, 48 expressed as Hex value

26, // length in octets, 38expressed as Hex value

01 00, // Version 1

00 0F AC 04, // CCMP as data group cipher suite

01 00, // pairwise cipher suite count

00 0F AC 04, // CCMP as pairwise cipher suite

01 00, // authentication count

00 0F AC 01 // IEEE 802.1X authentication

01 00 // Preauthentication capabilities

01 00 // PMKID Count

01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 // PMKID

Insert the following text before 7.3.2.25.1:

IEEE 802.1X authentication, CCMP pairwise and group key cipher suites (WEP-40, WEP-104, and TKIP are not allowed), and Robust Management Frame Protection are allowed and enforced with AES-128-CMAC as the broadcast/multicast management suite selector.

30, // information element id, 48 expressed as Hex value

1A, // length in octets, 26expressed as Hex value

01 00, // Version 1

00 0F AC 04, // CCMP as the data group key cipher suite

01 00, // pairwise key cipher suite count

00 0F AC 04, // CCMP as pairwise cipher suite

01 00, // authentication count

00 0F AC 01 // IEEE 802.1X authentication

80 00 // Robust Management Frame Protection is enabled

00 00 // No PMKIDs

00 0F AC <ANA>, // AES-128-CMAC as the broadcast/multicast management cipher suite

EDITORIAL NOTE : <ANA> request to ANA for assignment of AES-128-CMAC.

7.3.2.25.1 Cipher suites

Change the 1st paragraph of 7.3.2.25.1 as follows:

The Data Group Cipher Suite field contains the cipher suite selector used by the BSS to protect broadcast/multicast data frames~~traffic~~.

Insert a new paragraph after the 3rd paragraph of 7.3.2.25.1 as follows:

The Management Group Cipher Suite field contains the cipher suite selector used by the BSS to protect broadcast/multicast management frames.

Change Table 32 as follows:

Table 7-32—Cipher suite selectors

OUI	Suite Type	Meaning
00-0F-AC	0	Use group cipher suite
00-0F-AC	1	WEP-40
00-0F-AC	2	TKIP
00-0F-AC	3	Reserved
00-0F-AC	4	CCMP - default pairwise cipher suite in an RSNA
00-0F-AC	5	WEP-104
00-0F-AC	<ANA>	AES-128-CMAC - default management group cipher suite in a BIP enabled RSNA
00-0F-AC	?-255	Reserved
Vendor OUI	Other	Vendor specific
Other	any	Reserved

EDITORIAL NOTE: Last assigned value is 5; should request ANA for value 6, but leave as <ANA> as noted above.

Insert the following paragraph after the third paragraph of 7.3.2.25.1:

When Robust Management Frame Protection is enabled, the negotiated pairwise cipher suite is used to protect unicast Robust Management frames and the management group cipher suite is used to protect broadcast/multicast Robust Management frames. Use of AES-128-CMAC is only valid as a management group cipher suite.

Change Table 33 by adding two new columns “Unicast Robust Management ~~Frames~~ frames and “Broadcast/multicast Robust Management ~~Frames~~frames” under a general column heading “Enabled Robust Management Frame Protection” on the right and appending a new row for AES-128-CMAC with the new cell values as follows:

Table 7-33—Cipher suite usage

			Management Frame Protection enabled	
Cipher Suite Selector	**GTK**	**PTK**	**Unicast Robust Management frames**	**Broadcast/multicast Robust Management frames**
Use group key	No	Yes	No	No
WEP-40	Yes	No	No	No
WEP-104	Yes	No	No	No
TKIP	Yes	Yes	No	No
CCMP	Yes	Yes	Yes	No
AES-128-CMAC	No	No	No	Yes

7.3.2.25.3 RSN capabilities

Change Figure 7-74 with the following (change being the addition of bit 6 as Robust Management Frame Protection and changing “Reserved” to be 7):

B0	B1	B2 – B3	B4 – B5	B6	B7	B8	B9	B10	B11	B12-15
Pre-Auth	No Pairwise	PTKSA Replay Counter	GTKSA Replay Counter	Reserved	Robust Management Frame ~~protection~~Protection	Reserved	Peer-Key Enabled	SPP A-MSDU Capable	SPP A-MSDU Required	Reserved

Figure 7-74 — RSN Capabilities field format

EDITORIAL NOTE: This update requests TGw use bit 7 for Robust Management Frame Protection pending ANA assignment. This is realizing TGr (as of Draft 7.0) no longer uses bit 6 and but is currently assigned to TGr per ANA. Consider the above TBD pending ANA assignment.

Insert after DashList item “Bits 4-5”:

— Bit 6: Reserved.

— Bit 7: Robust Management Frame Protection. A STA sets this bit to 1 to enable protection of Robust Management Frames. If an AP sets `dot11RSNAUnprotectedManagementFramesAllowed` to then that AP allows RSNA connections from non-AP STAs which do not provide Management Frame Protection.

Change DashList item "Bits 6-8 and 10-15" as follows:

— Bit~~s~~ ~~6-~~8 and 12-15: Reserved. The remaining subfields of the RSN Capabilities field are reserved and shall be set to 0 on transmission and ignored on reception.

7.3.2.46 Fast BSS transition information element

EDITORIAL NOTE: This clause is introduced by TGr and is tracked by TGw.

Insert the following row (ignoring the header row) in Table 7-43d - "Sub-element IDs" before the "Reserved" entry and renumbering the "Reserved" values as appropriate:

Table 7-43m—Sub-element IDs

Value	Contents of data field	Length (in octets)
4	IGTK	24

Insert the following paragraph and Figure 7-95aj after the paragraph ending with the sentence " It is encoded following the conventions from 7.1.1" in 7.3.2.46:

IGTK contains the Integrity GTK, used for Robust Management frames. It is encoded in the same way as the GTK, as specified above. The IGTK sub-element format is shown in Figure 7-95aj.

	Sub-element ID	Length	KeyID	PN	Key	ICV (see 11A.8.5)
Octets	1	1	2	6	16	8

Figure 7-95aj—IGTK sub-element format

EDITORIAL NOTE: 802.11-2007 ends with 7.3.2.35, TGk adds it thru 41, TGr adds it through 48, TGn succeeds it through 53, TGw follows with 54

Insert at the end of sub clause 7.3.2.53 the new sub clause 7.3.2.54 as follows:

7.3.2.54 Management MIC information element

The Management MIC information element (MMIE) provides message integrity and protects broadcast/multicast Robust Management ~~Frames~~ frames from forgery and replay. Figure 7-95ak shows the MMIE format.

	Element ID	Length	Key ID	Sequence number	MIC
Octets	1	1	2	6	8

Figure 7-95ak —Management MIC information element format

The value of the Element ID field is TBD.
EDITORIAL NOTE : TBD request to ANA for assignment.

The Length field denotes the number of octets in the information element and has a value of 16.

The Key ID field identifies the IGTK used to compute the MIC. Bits 0-11 defines a value in the range 0-4095. Bits 12 - 15 are reserved and set to 0 on transmission and ignored on reception. The IGTK Key ID is either 4 or 5. The remaining Key IDs are reserved for future multicast extensions.

The Sequence Number field contains a 6 octet value, interpreted as a 48-bit unsigned integer and used to prevent replay of broadcast/multicast Robust Management frames.

The MIC field contains a message integrity code calculated over the Robust Management ~~Frame~~ frame as specified in 8.3.4.5 and 8.3.4.6.

8. Security

8.1 Framework

8.1.1 Security methods

Insert the following sub-item between "CCMP and RSNA" in 8.1.1:

— BIP, described in 8.3.4

8.1.2 RSNA equipment and RSNA capabilities

8.1.3 RSNA establishment

Insert sub-item '7' in the first item ('a') as follows:

7) If Robust Management Frame Protection is enabled, it programs the TK and pairwise cipher suite into the MAC for protection of robust unicast management frames. It also installs the IGTK, and IGTK sequence counter.

Insert sub-item '6' in the second item ('b') as follows:

6) If Robust Management Frame Protection is enabled, it programs the negotiated pairwise cipher suite and established PTK, IGTK, and IGTK sequence counter.

Change the title of 8.3 as follows:

8.3 RSNA ~~data~~ confidentiality and integrity protocols

8.3.1 Overview

Change the 1st paragraph of 8.3.1 as follows:

This standard defines two RSNA confidentiality and integrity protocols: TKIP and CCMP. This standard defines one integrity protocol: BIP.

Implementation of CCMP shall be mandatory in all IEEE 802.11 devices claiming RSNA compliance.

Implementation of TKIP is optional for an RSNA and used only for the protection of data frames. A design aim for TKIP was that the algorithm should be implementable within the capabilities of most devices supporting only WEP, so that many such devices would be field-upgradeable by the supplier to support TKIP.

BIP is a mechanism used only when protection of Robust Management frames is enabled and is used to provide integrity protection for broadcast/multicast Robust Management frames.

EDITORIAL NOTE : The updates to the above paragraph does not show the added paragraph breaks with underline/ strikethrough.

8.3.3 CTR with CBC-MAC Protocol (CCMP)

8.3.3.1 CCMP Overview

Insert the following paragraph at the end of 8.3.3.1:

When CCMP is selected as the RSN pairwise cipher and Management Frame Protection is enabled, unicast Robust Management frames shall be protected with CCMP. ~~A MAC implementation shall support CCMP for protecting management frames if CCMP and Management Frame Protection are both supported.~~

8.3.3.3 CCMP cryptographic encapsulation

8.3.3.3.2 Construct AAD

EDITORIAL NOTE: 8.3.3.3.2 is updated per TGn changes as well.

Replace Figure 8-17 with the following figure removing the muted bit descriptions and including the underlined updates:

	FC ~~(bits 4,5,6,11,12,13, 15= 0)~~ ~~(bit 14=1)~~	A1	A2	A3	SC ~~(bits 4-15=0)~~	A4	QC ~~(bits 4-6, 8-15 = 0)~~ (bit 7: see NOTE)
Octets	2	6	6	6	2	6	2

	FC ~~(bits 4,5,6,11,12,13= 0)~~ ~~(bit 14=1)~~	A1	A2	A3	SC ~~(bits 4-15=0)~~	A4	QC ~~(bits 4-15 = 0)~~
Octets	2	6	6	6	2	6	2

Figure 8-17—AAD construction

Change the third paragraph of 8.3.3.3.2 as follows:

The AAD is constructed from the MPDU Header. The AAD does not include the header Duration field, because the Duration field value can change due to normal IEEE 802.11 operation (e.g. a rate change during retransmission). The AAD does not include the Duration/ID field or the HT Control field, because the contents of these fields can change during normal operation (e.g., due to a rate change preceding re-transmission). The HT Control field can also be inserted or removed during normal operation (e.g., retransmission of an A-MPDU, where the original A-MPDU included an MCS request that has already generated a response). For similar reasons, several sub-fields in the Frame Control field are masked to 0. AAD construction is performed as follows:

a) FC - MPDU Frame Control field, with:
   1) Subtype bits (bits 4 5 6) in a Data MPDU masked to 0
   2) Retry bit (bit 11) masked to 0
   3) PwrMgt bit (bit 12) masked to 0
   4) MoreData bit (bit 13) masked to 0
   5) ~~The~~ Protected Frame bit (bit 14) always set to 1
   6) Order bit (bit 15) masked to 0

b) A1 - MPDU Address 1 field

c) A2 - MPDU Address 2 field

d) A3 - MPDU Address 3 field

e) SC - MPDU Sequence Control field, with the Sequence Number subfield (bits 4-15 of the Sequence Control field) masked to 0. The Fragment Number subfield is not modified.

f) A4 - MPDU Address field, if present~~ in the MPDU~~.

g) QC - QoS Control field, if present, a 2-octet field that includes the MDSU priority. The QC TID field is used in the construction of the AAD. When both the STA and its peer have their SPP A-MSDU Capable fields set to 1, bit 7 (the A-MSDU Present field) is used in the construction of the AAD. The remaining QC fields are set to 0 for the AAD calculaltion (bits 4 to 6, bits 8 to 15, and bit 7 when either the STA or its peer has the SPP A-MSDU Capable field set to 0).

8.3.3.3.3 Construct CCM nonce

Replace Figure ~~136~~ 8-18 with the following figure replacing "Priority Octet" with "Nonce Flag Octet", and addition of "Management" as bit 4 of this octet.":

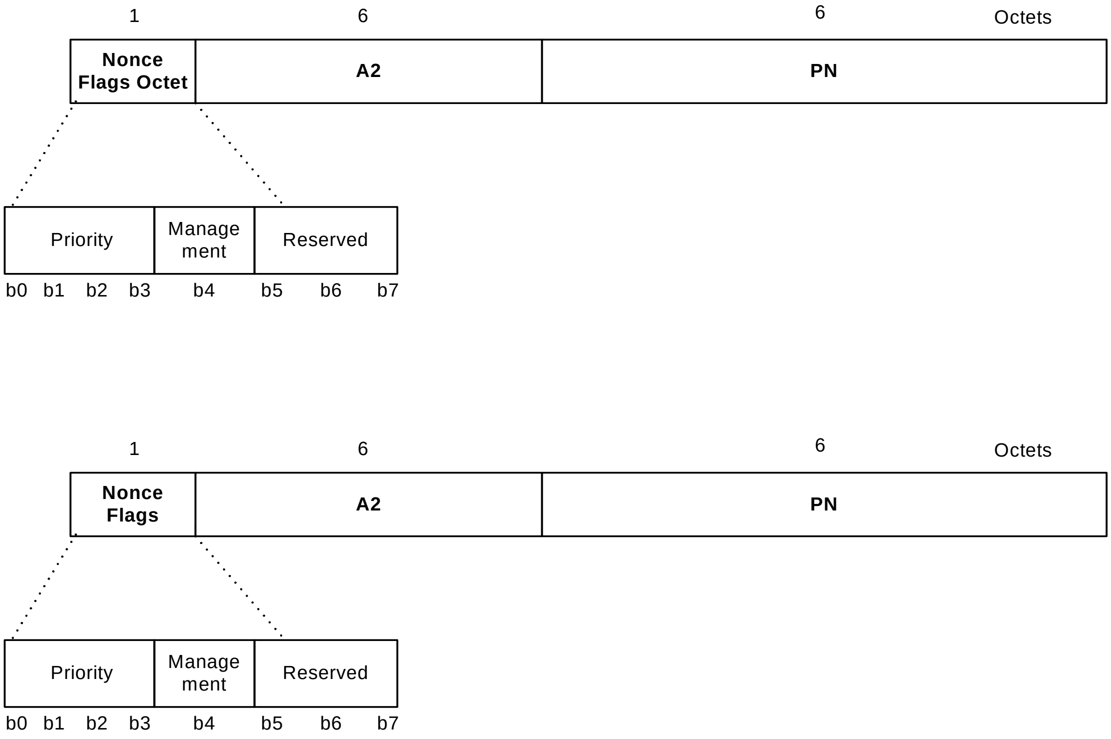


Figure 8-18 — Nonce Construction

EDITORIAL NOTE: ANA must be requested for this bit (4). There have been issues editing the above Figure and thus is left as B4....readers should note that until ANA assigns this bit, it should read <ANA>, pending ANA assignment.

Change the second paragraph 8.3.3.3.3 as follows:

The Nonce field has an internal structure of the ~~Priority~~ Nonce Flags ~~Octet~~ || A2 || PN (“||” is concatenation), where

- ~~The Priority Octet field shall be set to the fixed value 0 (0x00) when there is no QC field present in the MPDU header. When the QC field is present, bits o to 3 of the priority Octet field shall be set to the value of the QC TID (bits 0 to 3 of the QC field). Bits 4 to 7 of the Priority OCtet field are reserved and shall be set to 0.~~
- The Priority sub-field of the Nonce Flags ~~Octet~~ field shall be set to the fixed value 0 when there is no QC field present in the MPDU header. When the QC field is present, bits 0 to 3 of the Priority field shall be set to the value of the QC TID (bits 0 to 3 of the QC field).
- The Management field of the Nonce Flags field shall be set to 1 if the Type field of the Frame Control field is 00 (Management frame); otherwise it is set to 0.
- Bits 5 to 7 of the Nonce Flags field are reserved and shall be set to 0 on ~~transmission and ignored on reception~~transmission.
- MPDU Address A2 field occupies octets 1-6. This shall be encoded with the octets ordered with A2 octet 0 at octet index 1 and A2 octet 5 at octet index 6. If the frame is of type Management Frame and Management Frame protection is enabled, the MMPDU SA shall be encoded with the octets ordered with the MMPDU SA octet 0 at octet index 1 and MMPDU SA octet 5 at octet index 6.
- The PN field occupies octets 7-12. The octets of PN shall be ordered so that PN0 is at octet index 12 and PN5 is at octet index 7.

8.3.3.3.5 CCM originator processing

Insert the following text at the end of 8.3.3.3.5:

A CCMP protected unicast Robust Management frame shall be protected with the TK.

8.3.3.4 CCMP decapsulation

Change item 'c' as follows:

c) The Nonce value is constructed from the A2, PN, and ~~Priority~~ Nonce Flags ~~Octet~~ fields.

Insert the following paragraph before 8.3.3.4.1:

When the received frame is a CCMP protected robust unicast management frame, contents of the MMPDU body after protection is removed shall be delivered to the SME via the MLME primitive designated for that management frame rather than through the MA-UNITDATA.indication primitive.

8.3.3.4.1 CCM recipient processing

Insert the following sentence at the end of the first paragraph in 8.3.3.4.1:

A CCMP protected unicast Robust Management frame shall use the same TK as a Data MPDU.

8.3.3.4.2 Decrypted CCMP MPDU

8.3.3.4.3 PN and replay detection

Change item 'e' as follows:

e) For each PTKSA, GTKSA, and STKSA, the recipient shall maintain a separate replay counter for each IEEE 802.11 MSDU or A-MSDU priority and shall use the PN recovered from a received frame to detect replayed frames, subject to the limitation of the number of supported replay counters indicated in the RSN Capabilities field (see 7.3.2.25). A replayed frame occurs when the PN extracted from a received frame is less ~~that~~ than or equal to the current replay counter value for the frame's MSDU priority and frame type. A transmitter shall not use IEEE 802.11 MSDU or A-MSDU priorities without ensuring that the receiver supports the required number of replay counters. The transmitter shall not reorder frames within a replay counter, but may reorder frames across replay counters. One possible reason for reordering frames is the IEEE 802.11 MSDU or A-MSDU priority.

For each IGTKSA the recipient shall maintain a single frame replay counter for broadcast/multicast Robust Management ~~Frames~~frames, and shall use the PN recovered from received broadcast/multicast Robust Management Frames to detect replayed frames as described above for data frames

Insert the following bullet after 'e' :

e1) If `dot11RSNAProtectedManagementFramesEnabled` is TRUE, the recipient shall maintain a single management frame replay counter and shall use the PN from a received management frame to detect replayed management frames. A replayed frame occurs when the PN from a received management frame is less than or equal to the current management frame replay counter value. A replayed frame shall be silently discarded and the `dot11RSNAStatsRobustMgmtCCMPReplays` shall be incremented by 1. The transmitter shall preserve the order of Robust Management ~~Frames~~ frames sent to the same DA.

Insert the following after 8.3.3:

8.3.4 The Broadcast/Multicast integrity protocol

Broadcast/Multicast Integrity Protocol (BIP) provides data integrity and replay protection for broadcast/ multicast Robust Management frames after successful completion of either a 4-way Handshake or FT 4-way handshake, and delivery of the IGTK.

8.3.4.1 BIP overview

BIP provides data integrity and replay protection, using AES-128 in CMAC Mode. NIST SP 800-38B defines the CMAC algorithm. All BIP processing uses AES with a 128-bit integrity key and a 128-bit block size, and a CMAC TLen value of 64 (8 octets).

BIP uses the Integrity GTK (IGTK) to compute the broadcast/multicast MMPDU MIC. The authenticator shall distribute one new IGTK and IGTK PN whenever it distributes a new GTK. The IGTK is identified by the MAC address of the transmitting STA, plus ~~a non-zero 12-bit key~~ an IGTK identifier that is encoded in the MMIE Key ID field.

8.3.4.2 BIP MMPDU format

The Management MIC IE shall follow all of the other IEs in the management frame body but precede the FCS. See 7.3.2.54 for the format of the Management MIC IE. Figure 8-19a shows the BIP MMPDU.

IEEE 802.11 Header	Management Frame Body	Management MIC IE	FCS

IEEE 802.11 Header	Management Frame Body	Management MIC IE	FCS

Figure 8-19a — BIP Encapsulation

8.3.4.3 BIP AAD construction

The BIP Additional Authenticated Data (AAD) shall be constructed from the ~~MMPDU~~ MPDU header. The Duration field in the AAD shall be masked to 0. The AAD construction shall use a copy of the IEEE 802.11 header without the SC field for the ~~MMPDU~~MPDU, with the following exceptions:

a) FC - ~~MMPDU~~ MPDU Frame Control field, with:

   1) Retry bit (bit 11) masked to zero;

   2) PwrMgt bit (bit 12) masked to zero;

   3) MoreData bit (bit 13) masked to zero;

b) A1 - ~~MMPDU~~ MPDU Address 1 field;

c) A2 - ~~MMPDU~~ MPDU Address 2 field;

d) A3 - ~~MMPDU~~ MPDU Address 3 field.

Figure 8-19b depicts the format of the AAD. The length of the AAD is 20 octets.

	FC (bits 11, 12, 13 = 0)	A1	A2	A3
Octets:	2	6	6	6

	FC (bits 11, 12, 13 = 0)	A1	A2	A3
Octets:	2	6	6	6

Figure 8-19b — AAD Construction

8.3.4.4 BIP replay protection

The MMIE Replay field represents a sequence number whose length is 6 octets.

The transmitter shall insert a monotonically increasing value into the MMIE Replay field. The receiver shall maintain a 48-bit replay counter for each IGTK. The replay counter shall be set to the value of the IPN in the IGTK KDE provided by the Authenticator in either the 4-way handshake or Group Key handshakes. The receiver shall interpret the MMIE Replay field as a 48-bit integer. It shall then compare this integer value against the replay counter for the IGTK identified by the MMIE Key ID field. If the integer value from the received MMIE Replay Field is less than or equal to the replay counter value for the IGTK, the receiver shall silently discard the frame and increment the `dot11RSNAStatsCMACReplays` counter by 1.

Note - when the IGTK PN space is exhausted, the choices available to an implementation are to replace the IGTK or to end communications.

8.3.4.5 BIP transmission

When a STA transmits a protected broadcast/multicast Robust Management frame it shall:

a) Select the appropriate key (IGTK) for the frame and construct the MMIE (see 7.3.2.54) with the MIC field masked to zero and the KeyID field set to the corresponding IGTK KeyID value. The transmitter shall select a valid transmit sequence number as given in 8.3.4.4 and insert this value into the MMIE Replay Counter field.

b) Compute AAD as specified in 8.3.4.3.

c) Compute AES-128-CMAC over the concatenation of (AAD || Management Frame Body || MMIE), and insert the 64-bit output into the MMIE MIC field.

d) Compose the broadcast/multicast Robust Management frame as the IEEE 802.11 header, management frame body, MMIE, and FCS.

e) Transmit the broadcast/multicast Robust Management frame.

8.3.4.6 BIP reception

When a STA receives a broadcast/multicast Robust Management frame protectd by BIP, it shall:

a) Identify the appropriate IGTK key and associated state based on the MMIE KeyID field.

b) ~~Execute the appropriate~~ Perform replay protection ~~scheme~~ as defined in 8.3.4.4. If the replay protection fails, the `dot11RSNAStatsCMACReplays` shall be incremented by 1 and the frame shall be discarded. If the replay protection succeeds, the receiver shall compute AAD for this management frame, as specified in 8.3.4.3.

c) ~~If the replay protection scheme succeeds, the receiver shall compute AAD for this management frame, as specified in 8.3.4.3.~~ The receiver shall extract and save the received MIC value, and compute the AES-128-CMAC over the concatenation of (AAD || Management Frame Body || MMIE) with the MIC field masked to zero in the MMIE. If the result does not match the recieved MIC value, then the receiver shall discard the frame and increment the dot11RSNAStatsCMACICVErrors counter by 1.

If ~~the result does not match the recieved MIC value, then the receiver shall silently discard the frame and increment the dot11RSNAStatsCMACICVErrors counter by 1. If~~ Management Frame Protection is enabled, broadcast/multicast ~~management~~ Robust Management frames that are received without BIP protection shall be ~~silently~~ discarded..

8.4 RSNA security association management

8.4.1 Security associations

8.4.1.1 Security association definitions

Change the second paragraph as follows:

A security association is a set of policy(ies) and key(s) used to protect information. The information in the security association is stored by each party of the security association, must be consistent among all parties, and must have an identity. The identity is a compact name of the key and other bits of security association information to fit into a table index or an MPDU. ~~There are four types of security associations supported by an RSN STA:~~The following types of security associations are supported by an RSN STA:

Insert the following item after the third dashed item (i.e. after GTKSA) in the list:

— IGTKSA: A result of a successful Group Key Handshake, successful 4-way Handshake, or the (Re)association Response message of the Fast BSS Transition protocol.

Insert the following as a new sub clause succeeding 8.4.1.1.3:

8.4.1.1.3a IGTKSA

When Management Frame Protection is enabled, a Non-AP STA’s SME creates an IGTKSA when it receives Message 3 of the 4-Way Handshake, the (Re)association Response message of the Fast BSS Transition protocol, or Message 1 of the Group Key Handshake. The IGTKSA is unidirectional and is created by the 4-way Handshake, Fast BSS Transition protocol handshake, or the Group Key Handshake. The Authenticator’s SME creates an IGTKSA when it changes the IGTK with all STAs to which it has a valid PTKSA.

An IGTKSA consists of the following elements:

— Direction vector (whether the IGTK is used for transmit or receive)

— KeyID

— IGTK

— Authenticator MAC address.

8.4.1.2 Security association life cycle

8.4.1.2.1 Security association in an ESS

Change item 'd' as follows:

d) The last step is key management. The authentication process creates cryptographic keys shared between the IEEE 802.1X AS and the STA. The AS transfers these keys to the AP, and the AP and STA use one key confirmation handshake, called the 4-Way Handshake, to complete security association establishment. The key confirmation handshake indicates when the link has been secured by the keys and is ready to allow normal data traffic and Robust Management ~~Frames~~frames.

Change the last sentence of the last paragraph as follows:

A STA's SME uses this primitve when it deletes a PTKSA, ~~or~~ GTKSA or IGTKSA.

8.4.3 RSNA policy selection in an ESS

Insert the following text before 8.4.3.1:

An RSNA-capable AP may choose to associate with RSNA STAs with or without the capability for Management Frame Protection set in the RSN information element, as set in the policy variable `dot11RSNAUnprotectedManagementFramesAllowed`. A STA may choose not to associate with an AP that does not advertise protection of Robust Management ~~Frames~~ frames in the RSN capabilities. When an RSNA STA tries to associate without Management Frame PManagement Frame Protectionrotection, the AP may reject the (Re)association if `dot11RSNAUnprotectedManagementFramesAllowed` is set to false. A non-AP STA may use `dot11RSNAUnprotectedManagementFramesAllowed` to decide whether to associate with an AP that does advertise Management Frame Protection. Table 8-1a details all the possibilities.

Table 8-1a—Robust Management frame selection in an ESS

AP's State	Non-AP STA's State	AP Action	STA/Supplicant Action
dot11RSNAProtectedManagementFramesEnabled = TRUE	RSN IE Robust Mgmt Frame protection subfield = 1	if STA's RSN IE advertises a different Management Group Cipher suite then AP shall reject (re)association request with Reason Code "Invalid management group cipher" else AP may accept (re)association request. The AP shall transmit and receive unicast Robust Management ~~Frames~~ frames protected by the data pairwise cipher suite, and broadcast/multicast Robust Management ~~Frames~~ frames protected by the Management Group Cipher suite.	if AP's RSN IE advertises a different Management Group Cipher suite then STA shall not (re)associate with this AP else STA may (re)associate with this AP. The STA shall transmit and receive unicast Robust Management ~~Frames~~ frames protected by the data pairwise cipher suite, and receive broadcast/multicast Robust Management protected by the Management Group Cipher suite.

Table 8-1a—Robust Management frame selection in an ESS

AP's State	Non-AP STA's State	AP Action	STA/Supplicant Action
dot11RSNAProtectedManagementFramesEnabled = true AND dot11RSNAUnprotectedManagementFramesAllowed = TRUE	RSN IE Robust Mgmt Frame protection subfield = 0	AP may (re)associate with this STA. The AP shall transmit and receive unicast Robust Management frames to and from this STA unprotected; the AP shall protect broadcast/multicast Robust Management frames using the Management Group Cipher suite.	The STA may associate with this AP with no Robust Management ~~Frame~~ frame Protection.
dot11RSNAProtectedManagementFramesEnabled =TRUE AND dot11RSNAUnprotectedManagementFramesAllowed = FALSE	RSN IE Robust Mgmt Frame protection subfield = 0	AP shall reject any (Re)associate Request from this STA with reason code "Invalid management group cipher"	The STA may try to reassociate with this AP.
RSN IE Robust Mgmt Frame protection subfield = 0	dot11RSNAProtectedManagementFramesEnabled =TRUE AND dot11RSNAUnprotectedManagementFramesAllowed = TRUE	The AP may associate withthis STA with no Robust Management frame protection	The STA may reassociate with this AP. The STA shall transmit and receive unicast and broadcast/multicast Robust Management frames unprotected.
RSN IE Robust Mgmt Frame protection subfield = 0	dot11RSNAProtectedManagementFramesEnabled = true AND dot11RSNAUnprotectedManagementFramesAllowed = false	The AP transmits management frames unprotected and will silently discard any protected management frames it recieves	The STA shall not (Re)associate with this AP

8.4.4 RSNA policy selection in an IBSS

Insert a new sub clause 8.4.4.2 as follows:

8.4.4.2 Robust Management ~~Frame~~ frame policy selection in an IBSS

Management Frame Protection is valid only if RSNA is selected to protect data messages and `dot11RSNAProtectedManagementFramesEnabled` is set to TRUE.

In an IBSS two 4-Way Handshakes exchange RSN information elements to establish the security of the link between two STAs. The Management Frame Protection capabilities are determined by the RSN information elements exchanged in the 4-Way Handshake initiated by the Authenticator of the STA with the larger MAC address. Table 8-1b details all of the possibilities, including the case when the STA with the larger MAC address does not support Management Frame protection.

Table 8-1b—Robust Management frame selection in an IBSS

STA's State	Peer STA's message contents	Local STA's Action
`dot11RSNAProtectedManagementFramesEnabled` = TRUE	RSN IE Robust Mgmt Frame protection subfield = 1	if Peer STA's message contents has an RSN IE that advertises a mismatching Management Group Cipher suite then The STA's state shall abort 4-Way Handshakes with the Peer STA else 4-Way Handshakes with the Peer STA may complete successfully. The STA's state shall transmit and receive unicast Robust Management ~~Frames~~ frames protected by the data pairwise cipher suite, and broadcast/multicast Robust Management ~~Frames~~ frames protected by the Management Group Cipher suite.

Table 8-1b—Robust Management frame selection in an IBSS

STA's State	Peer STA's message contents	Local STA's Action
dot11RSNAProtectedManagementFramesEnabled = true AND dot11RSNAUnprotectedManagementFramesAllowed = true	RSN IE Robust Mgmt Frame protection subfield = 0	The STA's state may complete 4-Way Handshake with the Peer STA. The STA's state shall transmit and receive unicast Robust Management ~~Frames~~ frames unprotected; the STA's state shall transmit broadcast/multicast Robust Management ~~Frames~~ frames protected using the Management Group Cipher suite, but receive broadcast/multicast Robust Management ~~Frames~~ frames from the Peer STA's message contents unprotected.
dot11RSNAProtectedManagementFramesEnabled = true AND dot11RSNAUnprotectedManagementFramesAllowed = false	RSN IE Robust Mgmt Frame protection subfield = 0	The STA's state shall abort any 4-Way Handshakes with this Peer STA's message contents
dot11RSNAProtectedManagementFramesEnabled = false or not implemented	Any	The STA shall transmit Robust Management frames without protection, shall discard any unicast protected Robust Management frames it receives, and shall ignore the MMIE in any broadcast/multicast protected Robust Management frame it receives.

8.4.9 RSNA key management in an IBSS

Change the text of 8.4.9 as follows:

To establish a security association between two STAs in an IBSS, each STA's SME must have an accompanying IEEE 802.1X Authenticator and Supplicant. Each STA's SME initiates the 4-Way Handshake from the Authenticator to the peer STA's Supplicant (see 8.4.7). Two separate 4-Way Handshakes are conducted.

The 4-Way Handshake is used to negotiate the pairwise cipher suites, as described in 8.4.4. The IEEE 802.11 SME configures the temporal key portion of the PTK into the IEEE 802.11 MAC. Each Authenticator uses the KCK and KEK portions of the PTK negotiated by the exchange it initiates to distribute its own GTK and if Management Frame Protection is enabled, its own IGTK. Each Authenticator generates its own GTK and if Management Frame Protection is enabled, its own IGTK, and uses either the 4-Way Handshake or the Group Key Handshake to transfer the GTK and if Robust Management ~~Frames~~ frames protection is enabled, the IGTK, to other STAs with whom it has completed a 4-Way Handshake. The pairwise key used between any two STAs shall be the pairwise key from the 4-Way Handshake initiated by the STA with the highest MAC address.

A STA joining an IBSS is required to adopt the security configuration of the IBSS, which includes the group cipher suite, pairwise cipher suite, ~~and~~ AKMP, and if Management Frame protection is enabled, Management Group Cipher Suite (see 8.4.4). The STA shall not set up a security association with any STA having a different security configuration. The Beacon and Probe Response frames of the various STAs within an IBSS must reflect a consistent security policy, as the beacon initiation rotates among the STAs.

A STA joining an IBSS shall support and advertise, in the Beacon frame, the security configuration of the IBSS, which includes the group cipher suite, advertised pairwise cipher suite, ~~and~~ AKMP, and if Robust Management Frame Protection is enabled, Management Group Cipher Suite (see 8.4.4). The STA may use the Probe Request frame to discover the security policy of a STA, including additional unicast cipher suites the STA supports. A STA shall ignore Beacon frames that advertise a different security policy.

8.4.10 RSNA security association termination

Change first paragraph as follows:

When a non-AP STA SME receives a successful MLME Association or Reassociation confirm primitive that is not part of a Fast BSS Transition or receives or invokes an MLME Disassociation or Deauthentication primitive, it will delete some security associations. Similarly, when an AP SME receives an MLME Association or Reassociation indication primitive that is not part of a Fast BSS Transition, or receives or invokes an MLME Disassociation or Deauthentication primitive, it will delete some security associations. In the case of an ESS the non-AP STA's SME shall delete the PTKSA, GTKSA, IGTKSA, SMKSA, and any STKSA, and the AP's SME shall delete the PTKSA, and invoke an STSL application teardown procedure for any of its STKSAs. An example of an STSL application teardown procedure is described in 11.7.3. In the case of an IBSS, the STA's SME shall delete the PTKSA and the receive GTKSA and IGTKSA. Once the security associations have been deleted, the SME then invokes MLME-DELETEKEYS.request primitive to delete all temporal keys associated with the deleted security associations.The IEEE 802.1X Controlled Port returns to being blocked. As a result, all data frames are unauthorized before invocation of an MLME-DELETEKEYS.request primitive.

Insert sub-clauses 8.4.11 and 8.4.12 as follows:

8.4.11 Protection of unicast/broadcast/multicast management frames

When Management Frame Protection is enabled and the 4-Way Handshake is completed successfully, all transmissions of Robust Management frames shall be protected. When Management Frame Protection is enabled on the receiver and advertised by the transmitter, all received broadcast/multicast Robust Management frames shall be discarded if a matching IGTK is not available or if the frame is unprotected.

NOTE- BIP does not provide protection against forgery by associated and authenticated non-AP STAs.

Protection of broadcast/multicast management Action frames shall be provided by a service in the MLME as described in 11.7.

8.4.12 Robust Management ~~Frame~~ frame Selection Procedure

If the AKM suite selected in the RSN IE is 00-0F-AC:1 or 00-0F-AC:2, then Management Frame Protection shall apply to Robust Management ~~Frames~~ frames after the RSNA PTK key establishment is completes successfully and after the GTK and IGTK have been delivered. All management frames sent by a STA before keys are installed shall be unprotected. If Management Frame Protection is negotiated, all Action frames received before keys are installed shall be discarded.

If the AKM suite selected in the RSN IE is 00-0F-AC:3 or 00-0F-AC:4, then Robust Management Frame Protection shall apply to Robust Management frames after the FT 4-way handshake or FT protocol has completed, and the GTK and IGTK have been delivered. All management frames sent or received by a STA before the keys are installed shall be unprotected.

8.5 Keys and key distribution

8.5.1 Key hierarchy

Change the first paragraph and its succeeding item list as follows:

RSNA defines ~~two~~ the following key hierarchies:

a) Pairwise key hierarchy, to protect unicast traffic

b) GTK, a hierarchy consisting of a single key to protect multicast and broadcast/multicast traffic

NOTE - Pairwise key support with TKIP or CCMP allows a receiving STA to detect MAC address spoofing and data forgery. The RSNA architecture binds the transmit and receive addresses to the pairwise key. If an attacker creates an MPDU with the spoofed TA, then the decapsulation procedure at the receiver will generate an error. GTKs do not have this property.

c) Integrity GTK (IGTK), a hierarchy consisting of a single key to provide integrity protection for broadcast and multicast Robust Management frames

8.5.1.3 Group key hierarchy

Insert a new sub clause after 8.5.1.3 as follows:

8.5.1.3a Integrity group key hierarchy

The Authenticator shall select the IGTK as a random value each time it is generated.

The Authenticator may update the IGTK for reasons such as:

a) The disassociation or deauthentication of a STA.

b) An event within the STA's SME which triggers a Group Key Handshake.

The EAPOL-Key state machines (see 8.5.5 and 8.5.6) configure the IGTK via the MLME-SET-KEYS.request primitive.

The IGTK sequence counter is used to provide replay protection.

Note that a STA that has left the group can forge frames as an ~~outsider~~ insider until the IGTK is updated.

8.5.2 EAPOL-Key frames

Insert the following row into Table 8-4 - KDE before the "Reserved" entry and update the numbering appropriately:

Table 8-4—KDE

OUI	Data Type	Meaning
00-0F-AC	<ANA>	IGTK KDE

EDITORIAL NOTE: 802.11-2007 has values assigned up through 8. ~~Values~~ Value assigned above as <ANA> and ~~are~~ is pending ANA request assignments. The Reserved field must be updated as appropriate.

Insert the following text and Figure 8-32a before the paragraph starting "The following EAPOL-Key frames are used to implement the three different exchanges:":

The format of the IGTK KDE is shown in Figure 8-32a. The IGTK Packet Number (IPN) corresponds to the last PN used by the broadcast/multicast transmitter, to be used by the receiver as the initial value for the BIP replay counter.

KeyID	IPN	IGTK
2 octets	6 octets	16 octets

Figure 8-32a—IGTK KDE format

8.5.2.1 EAPOL-Key frame notation

Insert the following text before the notation for "PMKID":

IGTK[M] is the IGTK, with key identifier field set to M.

IPN is the current IGTK replay counter value provided by the IGTK KDE

8.5.3 4-Way Handshake

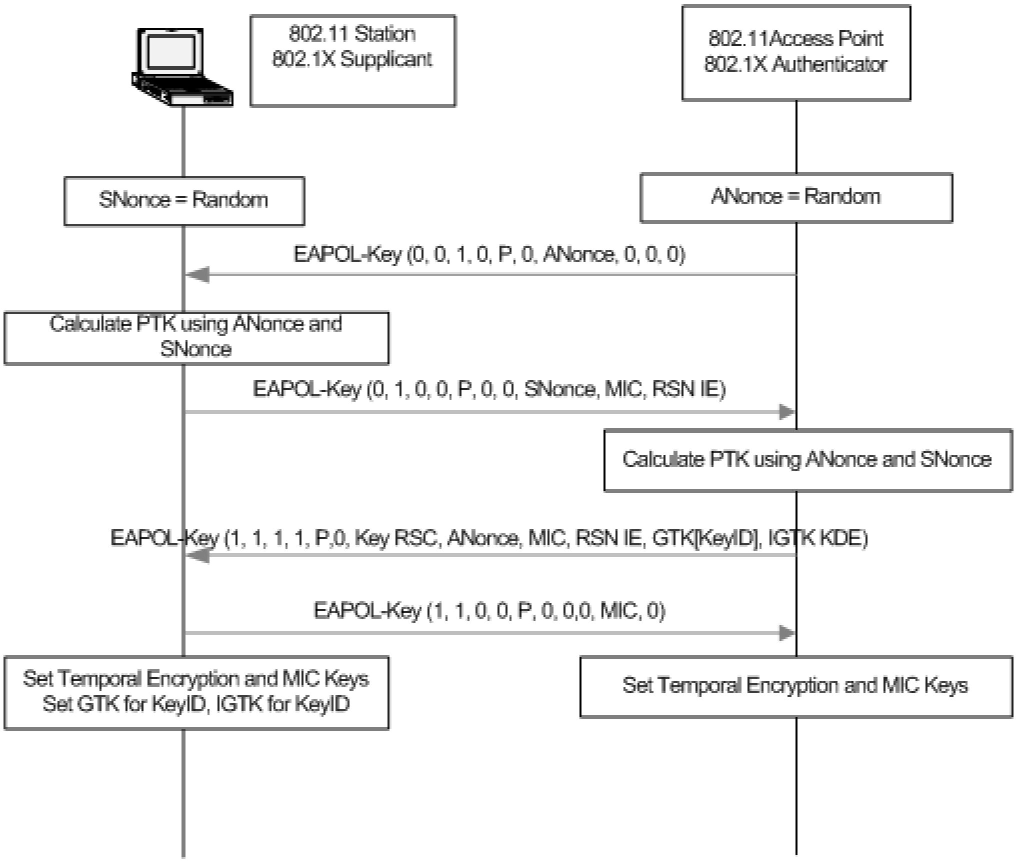
8.5.3.3 4-Way Handshake Message 3

Change the entry for "Key Data" in 8.5.3.3 as indicated below:

Key Data = For PTK generation, the AP's Beacon/Probe Response frame's RSN information element, and, optionally, a second RSN information element that is the Authenticator's pairwise cipher suite assignment, and, if a group cipher has been negotiated, the encapsulated GTK and the GTK's key identifier (see 8.5.2), and if Management Frame Protection is enabled, the IGTK KDE. For STK generation Initiator RSN IE, Lifetime of SMK is used.

8.5.3.6 Sample 4-Way Handshake

Replace Figure 151 with the following Figure, with the updates including IGTK KDE on the 3rd EAPOL-Key message and "IGTK for Key ID" on the 3rd supplicant box:


802.11 Station
802.1X Supplicant
802.11Access Point
802.1X Authenticator
SNonce = Random
ANonce = Random
EAPOL-Key (0, 0, 1, 0, P, 0, ANonce, 0, 0, 0)
Calculate PTK using ANonce and SNonce
EAPOL-Key (0, 1, 0, 0, P, 0, 0, SNonce, MIC, RSN IE)
Calculate PTK using ANonce and SNonce
EAPOL-Key (1, 1, 1, 1, P,0, Key RSC, ANonce, MIC, RSN IE, GTK[KeyID], IGTK KDE)
EAPOL-Key (1, 1, 0, 0, P, 0, 0,0, MIC, 0)
Set Temporal Encryption and MIC Keys
Set GTK for KeyID, IGTK for KeyID
Set Temporal Encryption and MIC Keys

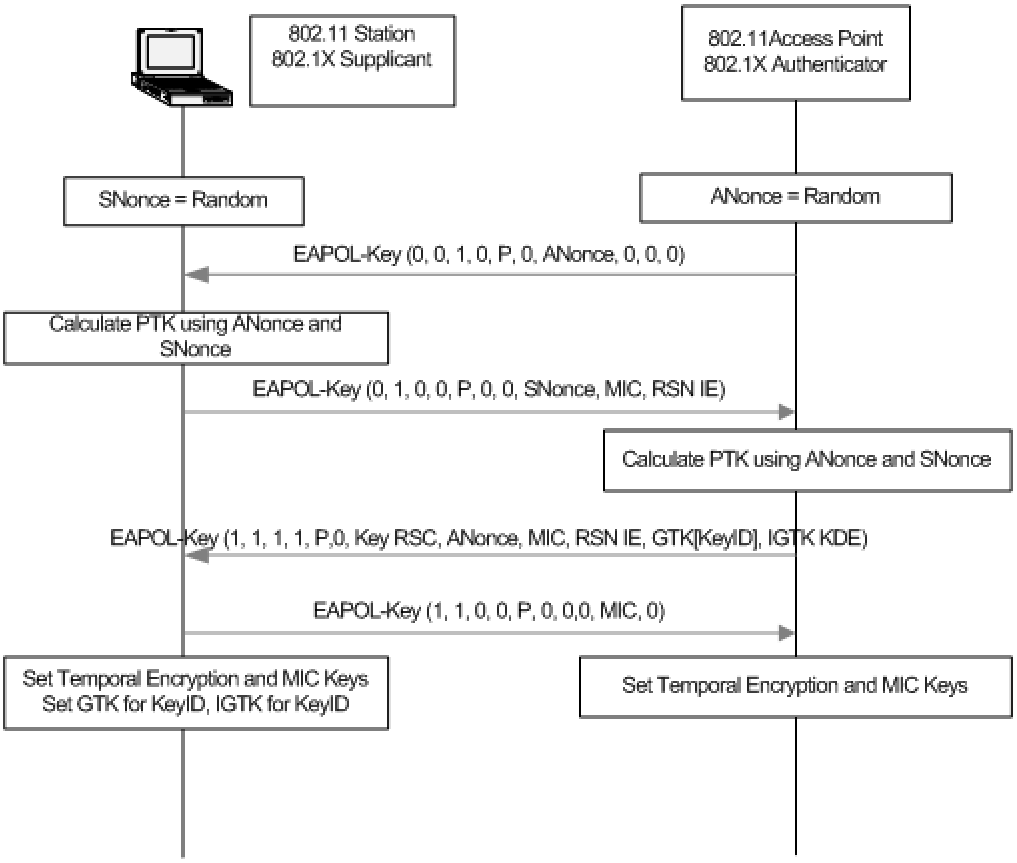

802.11 Station
802.1X Supplicant
802.11Access Point
802.1X Authenticator
SNonce = Random
ANonce = Random
EAPOL-Key (0, 0, 1, 0, P, 0, ANonce, 0, 0, 0)
Calculate PTK using ANonce and SNonce
EAPOL-Key (0, 1, 0, 0, P, 0, 0, SNonce, MIC, RSN IE)
Calculate PTK using ANonce and SNonce
EAPOL-Key (1, 1, 1, 1, P,0, Key RSC, ANonce, MIC, RSN IE, GTK[KeyID], IGTK KDE)
EAPOL-Key (1, 1, 0, 0, P, 0, 0,0, MIC, 0)
Set Temporal Encryption and MIC Keys
Set GTK for KeyID, IGTK for KeyID
Set Temporal Encryption and MIC Keys

Figure 8-33—Sample 4-Way Handshake

Change the text in 8.5.3.6 item 'e' as follows:

e) The Authenticator sends an EAPOL-Key frame containing ANonce, the RSN information element from its Beacon or Probe Response messages, MIC, whether to install the temporal keys, ~~and~~ the encapsulated GTK, and if Management Frame Protection is enabled, the IGTK.

8.5.4 Group Key Handshake

Change the text of the first 3 paragraphs including the itemized list as follows:

The Authenticator uses the Group Key Handshake to send a new GTK, and, if Management Frame protection is enabled, a new IGTK to the Supplicant.

The Authenticator may initiate the exchange when a Supplicant is disassociated or deauthenticated.

Message 1: Authenticator → Supplicant: EAPOL-Key(1,1,1,0,G,0,Key RSC,0, MIC, GTK[N], IGTK[M])

Message 2: Supplicant ←→ Authenticator: EAPOL-Key(1,1,0,0,G,0,0,MIC,0)

Here, the following assumptions apply:

— Key RSC denotes the last frame sequence number sent using the GTK.

— GTK[N] denotes the GTK encapsulated with its key identifier as defined in 8.5.2 using the KEK defined in 8.5.1.2 and associated IV.

— IGTK[M], when present, denotes the IGTK encapsulated with its key identifier as defined in 8.5.2 using the KEK defined in 8.5.1.2 and associated IV.

— The MIC is computed over the body of the EAPOL-Key frame (with the MIC field zeroed for the computation) using the KCK defined in 8.5.1.2.

8.5.4.1 Group Key Handshake Message 1

Change the description for 'Key Data' in 8.5.4.1 as follows:

Key Data = encrypted, encapsulated

- GTK and the GTK's key identifier (see 8.5.2)

- When present, IGTK, IGTK's key identifier, and IPN (see 8.5.2)

Change item 'c' in 8.5.4.1 as follows:

c) Uses the MLME-SETKEYS.request primitive to configure the temporal GTK and, when present, IGTK into its IEEE 802.11 MAC.

8.5.4.3 Group Key Handshake implementation considerations

Change the second paragraph as follows:

The state machines in 8.5.5 and 8.5.6 change the GTK and, when present, IGTK in use by the network. See Figure 152**.**

Change the last paragraph and its numbered list as follows:

The following steps occur:

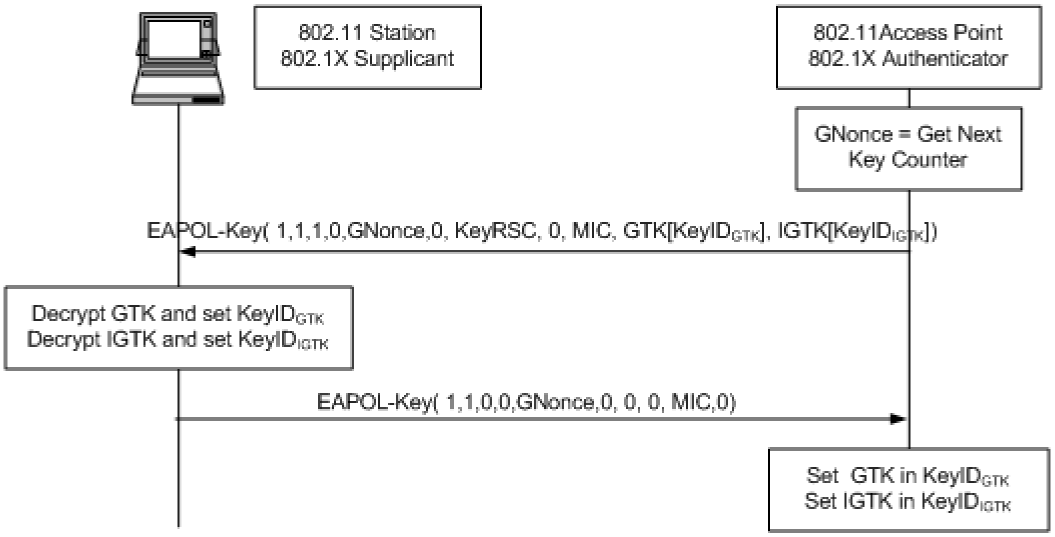
a) The Authenticator generates a new GTK and, when Robust Management frame protection has been negotiated, a new IGTK. It encapsulates the GTK and as necessary IGTK and sends an EAPOL-Key frame containing the GTK and IGTK (Message 1), along with the last sequence number used with the GTK (RSC) and the last sequence number used with the IGTK (IPN).

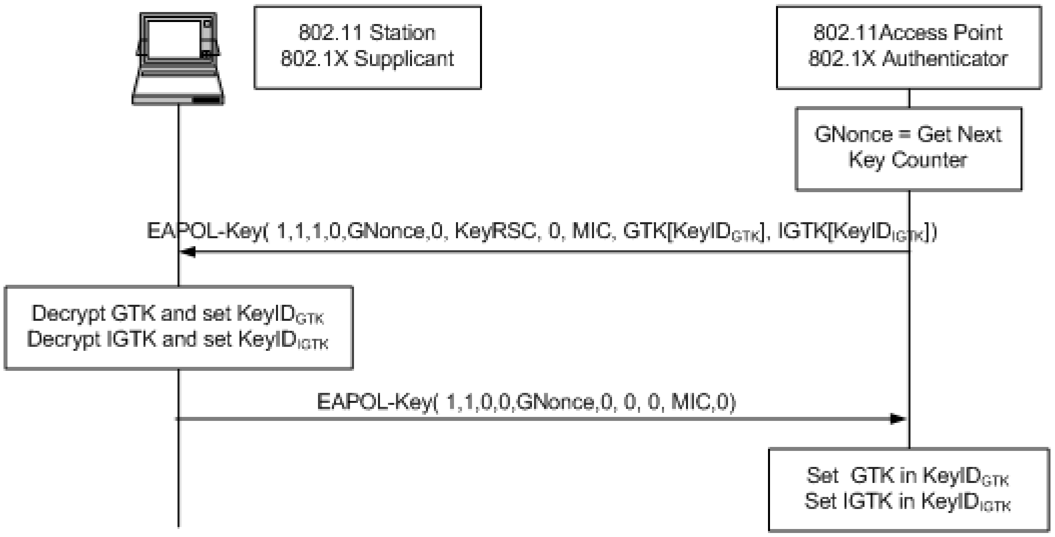
b) On receiving the EAPOL-Key frame, the Supplicant validates the MIC, decapsulates the GTK and, when present, the IGTK and uses the MLME-SETKEYS.request primitive to configure the GTK, IGTK, RSC, and IPN in its STA.

c) The Supplicant then constructs and sends an EAPOL-Key frame in acknowledgment to the Authenticator.

d) On receiving the EAPOL-Key frame, the Authenticator validates the MIC. If the GTK, and, if present, IGTK ~~is~~ are not already configured into IEEE 802.11 MAC, after the Authenticator has delivered the GTK, and IGTK to all associated STAs, it uses the MLME-SETKEYS.request primitive to configure the GTK, and IGTK into the IEEE 802.11 STA.

Replace Figure 8-34 with the following Figure, with the updates including IGTK on the first EAPOL-Key message and 2nd supplicant box, reformatting the first message above the message arrow and IGTK and new subscripts to the last Authenticator box:


802.11 Station
802.1X Supplicant
802.11Access Point
802.1X Authenticator
GNonce = Get Next Key Counter
EAPOL-Key(1,1,1,0,GNonce,0, KeyRSC, 0, MIC, GTK[KeyID$_{GTK}$], IGTK[KeyID$_{IGTK}$])
Decrypt GTK and set KeyID$_{GTK}$
Decrypt IGTK and set KeyID$_{IGTK}$
EAPOL-Key(1,1,0,0,GNonce,0, 0, 0, MIC,0)
Set GTK in KeyID$_{GTK}$
Set IGTK in KeyID$_{IGTK}$



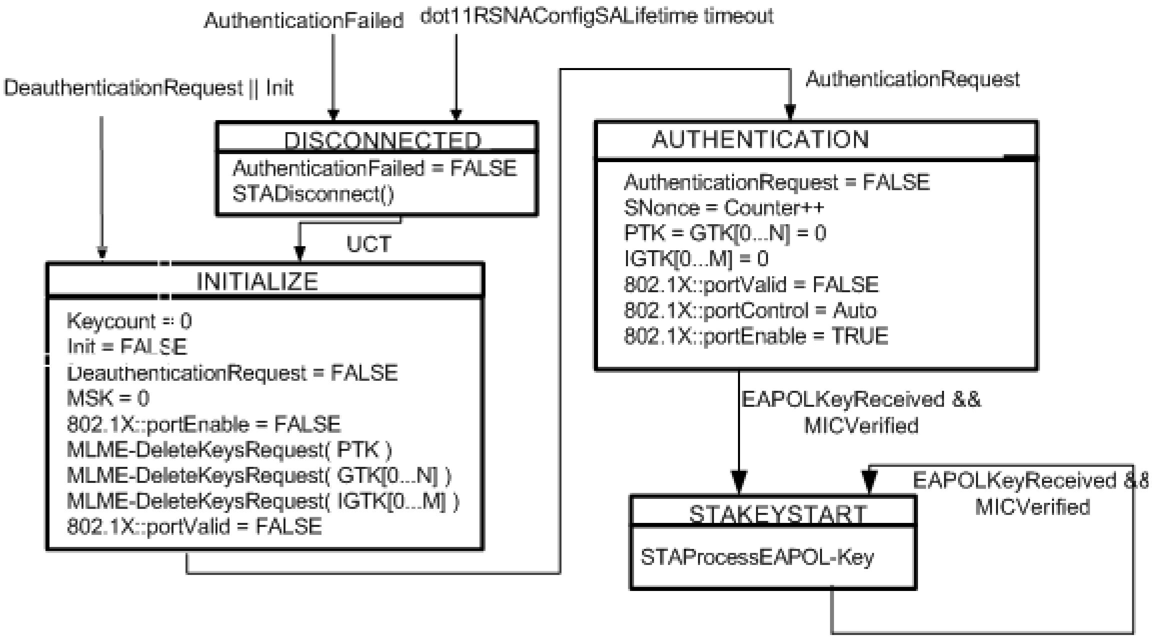
802.11 Station
802.1X Supplicant
802.11Access Point
802.1X Authenticator
GNonce = Get Next Key Counter
EAPOL-Key(1,1,1,0,GNonce,0, KeyRSC, 0, MIC, GTK[KeyID$_{GTK}$], IGTK[KeyID$_{IGTK}$])
Decrypt GTK and set KeyID$_{GTK}$
Decrypt IGTK and set KeyID$_{IGTK}$
EAPOL-Key(1,1,0,0,GNonce,0, 0, 0, MIC,0)
Set GTK in KeyID$_{GTK}$
Set IGTK in KeyID$_{IGTK}$

Figure 8-34—Sample Group Key Handshake

8.5.5 RSNA Supplicant key management state machine

8.5.5.1 Supplicant state machine states

Replace Figure 8-35 with the following figure updating "Snonce" to "SNonce", adding "IGTK[0...M] = 0" in the "AUTHENTICATION" box and "MLME-DeleteKeysRequest(IGTK[0...M])" to the "INITIALIZE" box:


AuthenticationFailed
dot11RSNAConfigSALifetime timeout
DeauthenticationRequest || Init
AuthenticationRequest
DISCONNECTED
AuthenticationFailed = FALSE
STADisconnect()
UCT
AUTHENTICATION
AuthenticationRequest = FALSE
SNonce = Counter++
PTK = GTK[0...N] = 0
IGTK[0...M] = 0
802.1X::portValid = FALSE
802.1X::portControl = Auto
802.1X::portEnable = TRUE
INITIALIZE
Keycount = 0
Init = FALSE
DeauthenticationRequest = FALSE
MSK = 0
802.1X::portEnable = FALSE
MLME-DeleteKeysRequest(PTK)
MLME-DeleteKeysRequest(GTK[0...N])
MLME-DeleteKeysRequest(IGTK[0...M])
802.1X::portValid = FALSE
EAPOLKeyReceived &&
MICVerified
EAPOLKeyReceived &&
MICVerified
STAKEYSTART
STAProcessEAPOL-Key

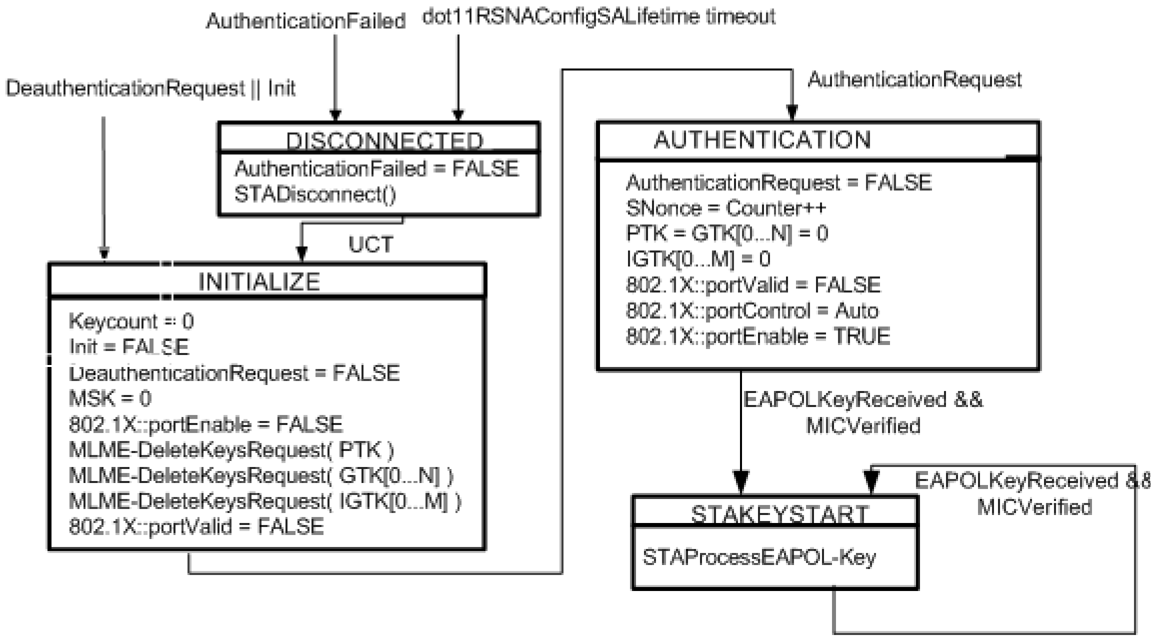

AuthenticationFailed
dot11RSNAConfigSALifetime timeout
DeauthenticationRequest || Init
AuthenticationRequest
DISCONNECTED
AuthenticationFailed = FALSE
STADisconnect()
UCT
AUTHENTICATION
AuthenticationRequest = FALSE
SNonce = Counter++
PTK = GTK[0...N] = 0
IGTK[0...M] = 0
802.1X::portValid = FALSE
802.1X::portControl = Auto
802.1X::portEnable = TRUE
INITIALIZE
Keycount = 0
Init = FALSE
DeauthenticationRequest = FALSE
MSK = 0
802.1X::portEnable = FALSE
MLME-DeleteKeysRequest(PTK)
MLME-DeleteKeysRequest(GTK[0...N])
MLME-DeleteKeysRequest(IGTK[0...M])
802.1X::portValid = FALSE
EAPOLKeyReceived && MICVerified
EAPOLKeyReceived &8 MICVerified
STAKEYSTART
STAProcessEAPOL-Key

Figure 8-35—RSNA Supplicant key management state machine

8.5.5.2 Supplicant state machine variables

Insert the following text immediately following the 'GTK[]' variable:

— IGTK[] - This variable represents the current IGTKs for each management group key index.

8.5.5.3 Supplicant state machine procedures

Change the 'StaProcessEAPOL-Key' item as follows:

— **StaProcessEAPOL-Key** - The Supplicant invokes this procedure to process a received EAPOL-Key frame. The pseudo-code for this procedure is as follows:

```
StaProcessEAPOL-Key (S, M, A, I, K, RSC, ANonce, RSC, MIC, RSNIE, GTK[N], IGTK[M], PN)
TPTK ← PTK
TSNonce ← 0
PRSC ← 0
UpdatePTK ← 0
State ← UNKNOWN
if M = 1 then
    if Check MIC(PTK, EAPOL-Key frame) fails then
        State ← FAILED
    else
        State ← MICOK
    endif
endif
```

```
if K = P then
    if State ~~!=~~ ≠ FAILED then
        if PSK exists then - PSK is a preshared key
            PMK ← PSK
        else
            PMK ← L(MSK, 0, 256)
        endif
    TSNonce ← SNonce
    if ANonce ~~!=~~ ≠ PreANonce then
        TPTK ← Calc PTK(PMK, ANonce, TSNonce)
        PreANonce ← ANonce
    endif
    if  State = MICOK then
        PTK ← TPTK
        UpdatePTK ← I
        if UpdatePTK = 1 then
            if no GTK then
                PRSC ← RSC
            endif
            if MLME-SETKEYS.request(0, TRUE, PRSC, PTK) fails then
                invoke MLME-DEAUTHENTICATE.request
            endif
            MLME.SETPROTECTION.request(TA, Rx)
        endif
        if GTK then
            if (GTK[N] ← Decrypt GTK) succeeds then
                if MLME-SETKEYS.request(N, 0, RSC, GTK[N]) fails then
                    invoke MLME-DEAUTHENTICATE.request
                endif
                else
                    State ← FAILED
                endif
        endif
        if IGTK then
            if (IGTK[M] ← Decrypt IGTK) succeeds then
                if MLME-SETKEYS.request(M, 0, PN, IGTK[M]) fails then
                    invoke MLME-DEAUTHENTICATE.request
                endif
                else
                    State ← FAILED
                endif
            endif
            endif
```

```
          endif
      else if KeyData = GTK then
          if State = MICOK then
             if (GTK[N] ← Decrypt GTK) succeeds then
               if MLME-SETKEYS.request(N, T, RSC, GTK[N]) fails then
                  invoke MLME-DEAUTHENTICATE request
               endif
               else
                  State ← FAILED
               endif
               if (IGTK[M] ← Decrypt IGTK) succeeds then
                  if MLME-SETKEYS.request(M, T, PN, IGTK[M]) fails then
                     invoke MLME-DEAUTHENTICATE request
                         endif
                  else
                     State ← FAILED
                  endif
               else
                  State ← FAILED
               endif
          endif
          if A = 1 && State ≠ Failed then
               MLME-SETPROTECTION.request(0,1,0,0,K,0,0,TSNonce,MIC(TPTK),RSNIE)
          endif
          if UpdatePTK = 1 then
               MLME-SETPROTECTION.request(TA, Tx_Rx)
          endif
          if State = MICOK && S = 1 then
               MLME-SETPROTECTION.request(TA, Tx_Rx)
               if IBSS then
                  keycount++
                  if keycount = 2 then
                  802.1X::portValid ← TRUE
                  endif
               else
                  802.1X::portValid ← TRUE
               endif
          endif
```

Change the second paragraph succeeding the pseudocode as follows:

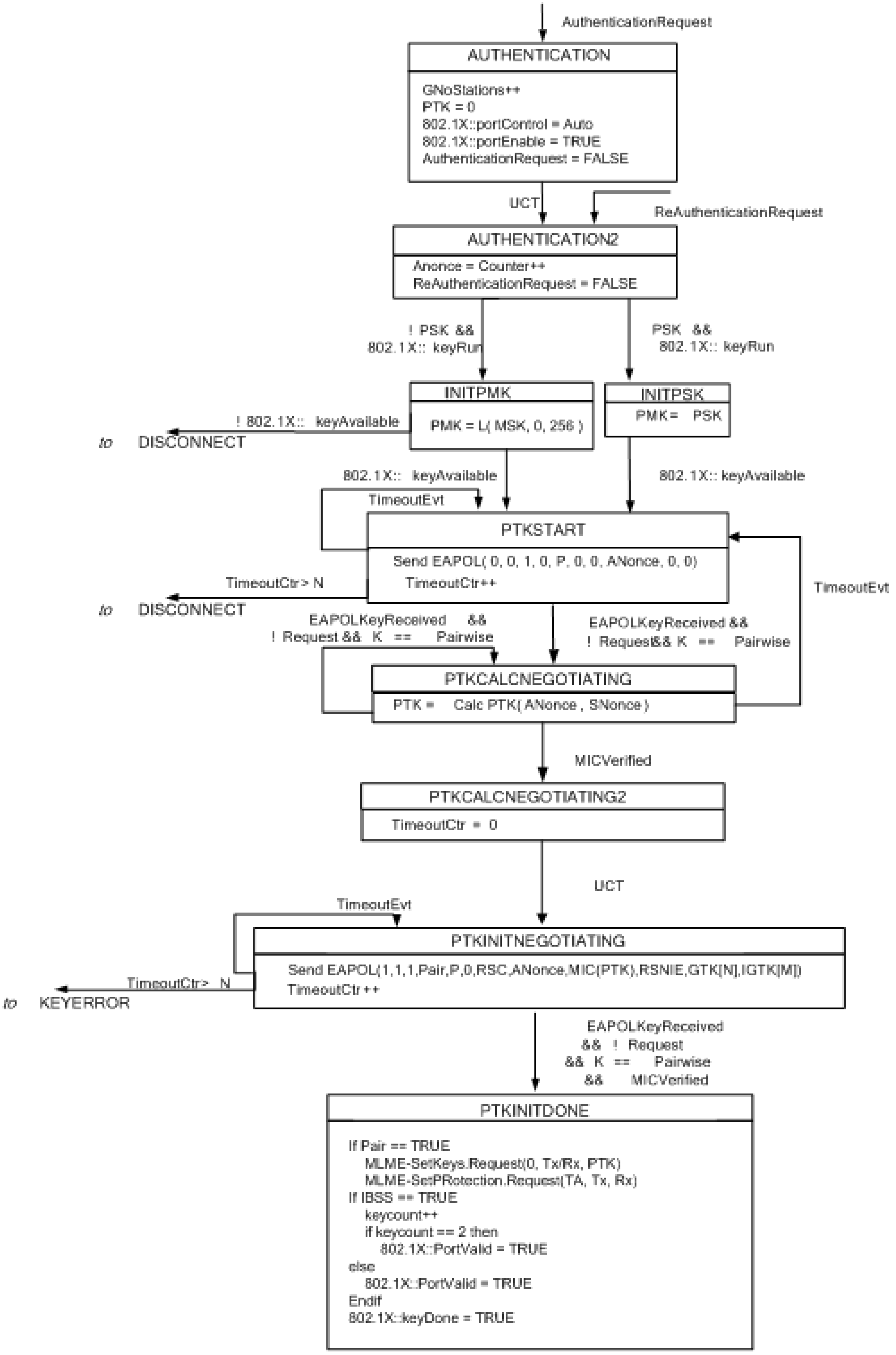
When processing 4-Way Handshake Message 3, the GTK and IGTK are ~~is~~ decrypted from the EAPOL-Key frame and installed. The PTK shall be installed before the GTK and IGTK.

Insert the following items at the end of 8.5.5.3:

— **DecryptIGTK(x)** - Decrypt the IGTK from the EAPOL-Key frame.

8.5.6 RSNA Authenticator key management state machines

Replace Figure 8-37 with the following Figure, with the updates being fixing Anonce with ANonce in the PTKSTART, PTKCALCNEGOTIATING, and PTKINITNEGOTIATING and the insertion of "IGTK[M]" in the PTKINITNEGOTIATING box:


AuthenticationRequest
AUTHENTICATION
GNoStations++
PTK = 0
802.1X::portControl = Auto
802.1X::portEnable = TRUE
AuthenticationRequest = FALSE
UCT
ReAuthenticationRequest
AUTHENTICATION2
Anonce = Counter++
ReAuthenticationRequest = FALSE
! PSK &&
802.1X:: keyRun
PSK &&
802.1X:: keyRun
INITPMK
PMK = L(MSK, 0, 256)
INITPSK
PMK = PSK
! 802.1X:: keyAvailable
to DISCONNECT
802.1X:: keyAvailable
802.1X:: keyAvailable
TimeoutEvt
PTKSTART
Send EAPOL(0, 0, 1, 0, P, 0, 0, ANonce, 0, 0)
TimeoutCtr++
TimeoutCtr > N
to DISCONNECT
TimeoutEvt
EAPOLKeyReceived &&
! Request && K == Pairwise
EAPOLKeyReceived &&
! Request&& K == Pairwise
PTKCALCNEGOTIATING
PTK = Calc PTK(ANonce , SNonce)
MICVerified
PTKCALCNEGOTIATING2
TimeoutCtr = 0
UCT
TimeoutEvt
PTKINITNEGOTIATING
Send EAPOL(1,1,1,Pair,P,0,RSC,ANonce,MIC(PTK),RSNIE,GTK[N],IGTK[M])
TimeoutCtr++
TimeoutCtr> N
to KEYERROR
EAPOLKeyReceived
&& ! Request
&& K == Pairwise
&& MICVerified
PTKINITDONE
If Pair == TRUE
MLME-SetKeys.Request(0, Tx/Rx, PTK)
MLME-SetPRotection.Request(TA, Tx, Rx)
If IBSS == TRUE
keycount++
if keycount == 2 then
802.1X::PortValid = TRUE
else
802.1X::PortValid = TRUE
Endif
802.1X::keyDone = TRUE

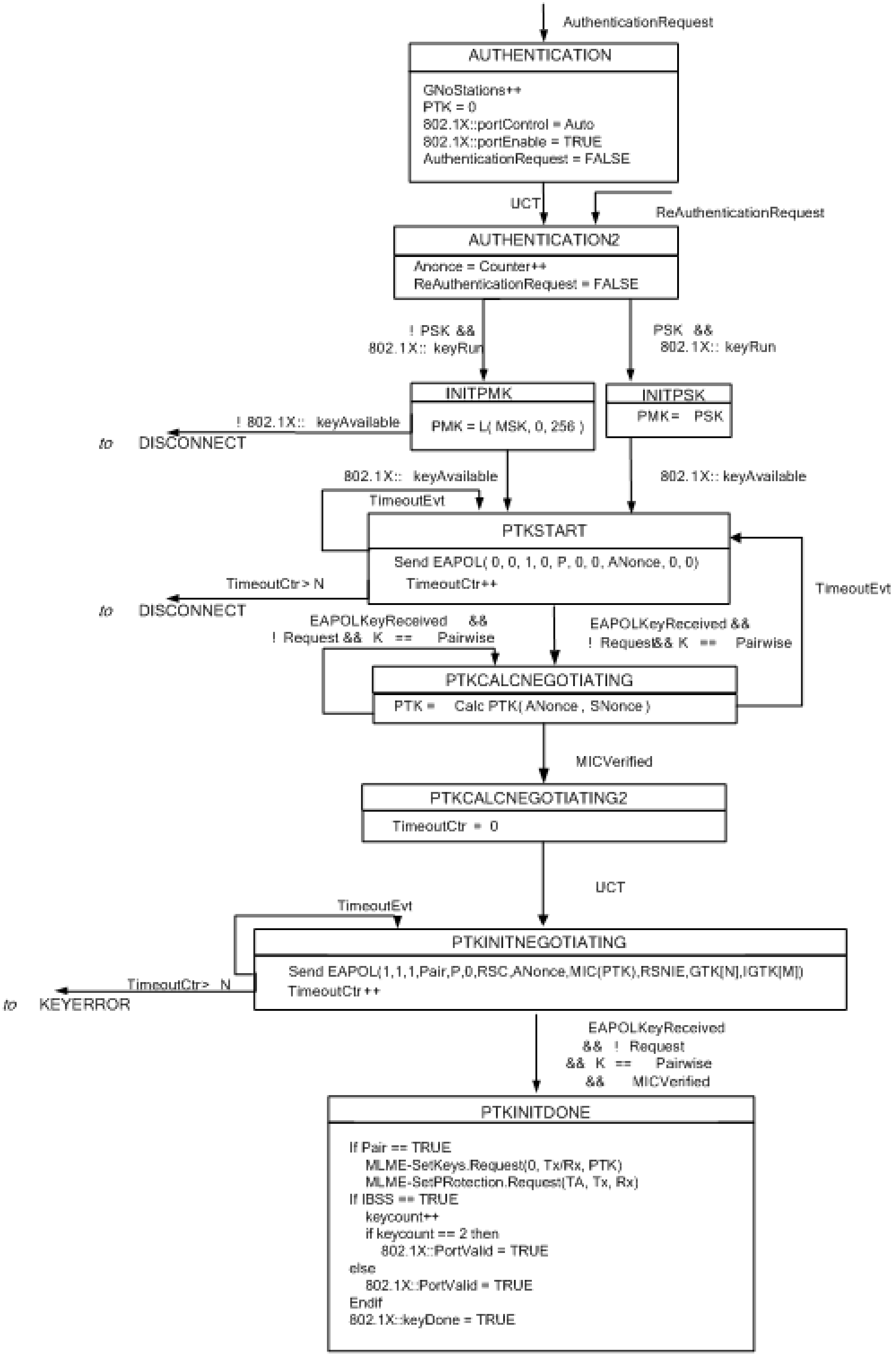
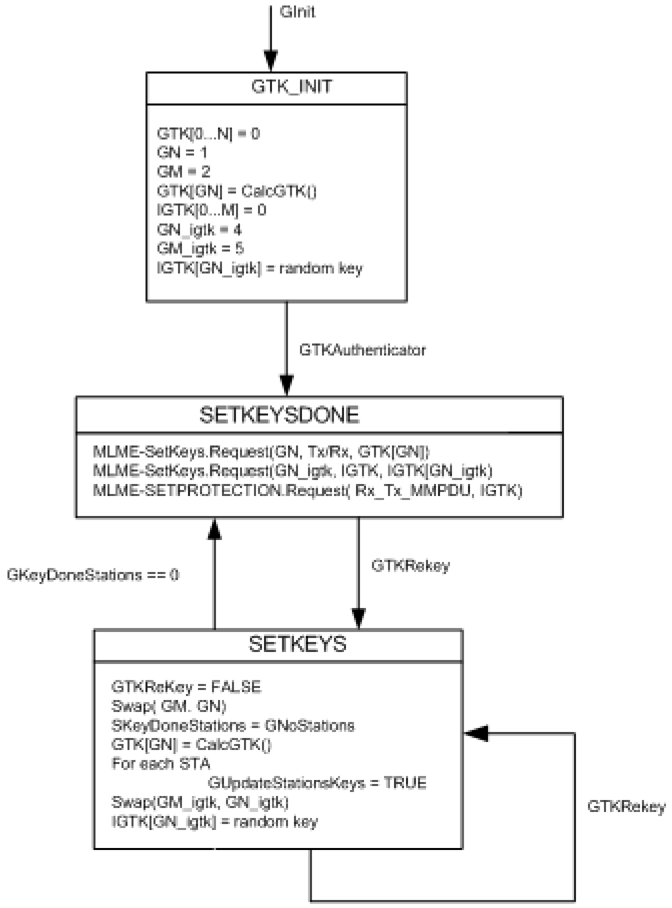


Figure 8-37—Authenticator state machines, part 1

Replace Figure 8-40 with the following Figure, with the updates being the addition into the GTK_INIT state (semicolons showing linebreaks): “IGTK[0..M] = 0; GN_igtk = 4; GM_igtk = 5; IGTK[GN_igtk] = random key”; addition into SETKEYSDONE state: “MLME-SETKEYS.request(GN_igtk, IGTK, IGTK[GN_igtk]); MLME-SETPROTECTION.request(Rx_Tx_MMPDU, IGTK)”. Addition into SET-KEYS state: “Swap(GM_igtk, GN_igtk); IGTK[GN_igtk] = random key”.insertion of “IGTK[M]” in the PTKINITNEGOTIATING box:



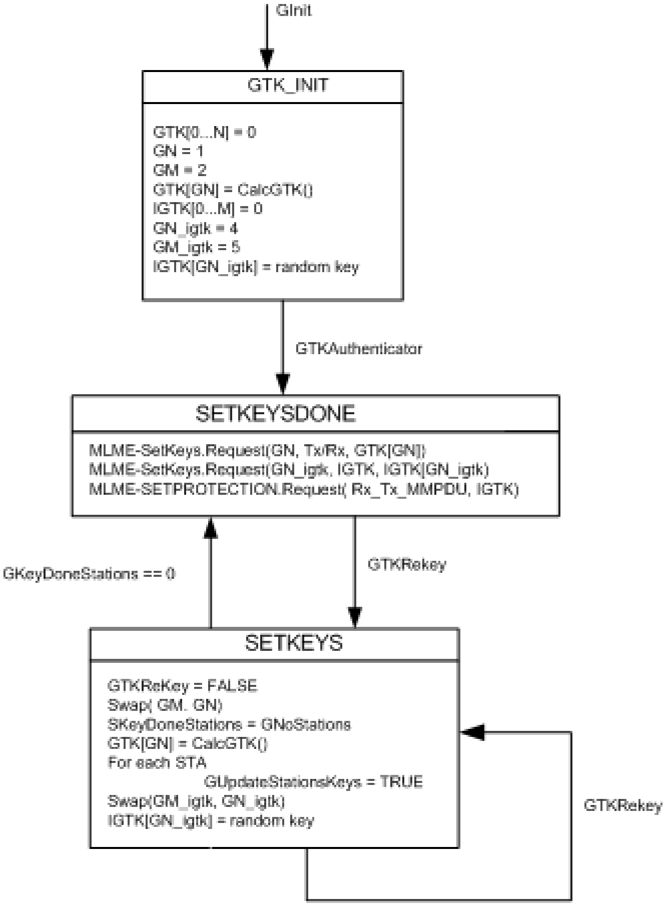


Figure 8-40—Authenticator state machines, part 4

8.6 Mapping EAPOL Keys to IEEE 802.11 keys

8.6.3 Mapping PTK to CCMP keys

Change the 2nd paragraph in 8.6.3 as follows:

A STA shall use the temporal key as the CCMP key for MSDUs and MMPDUs or A-MSDUs between the two communicating STAs.

Insert a new sub clause after 8.6.6 as follows:

8.6.6a Mapping IGTK to BIP Keys

See 8.5.1.3a for the definition of the IGTK key. A STA shall use bits 0-127 of the IGTK as the AES-128-CMAC key.

8.7 Per-frame pseudo-code

8.7.2 RSNA frame pseudo-code

Change the paragraph as follows:

STAs transmit protected MSDUs or MMPDUs to an RA when temporal keys are configured and an MLME~~.~~-SETPROTECTION.request primitive has been invoked ~~for transmit~~ with ProtectType parameter ~~Tx~~ Tx, Rx Tx, Tx MMPDU or ~~Rx Tx~~ Rx Tx MMPDU to that RA. STAs expect to receive protected MSDUs or MMPDUs and A-MSDUs from a TA when temporal keys are configured and an MLME~~.~~-SET-PROTECTION.request primitive has been invoked ~~for receive~~ with ProtectType parameter ~~Rx~~ Rx, Rx Tx, Tx MMPDU or ~~Rx_Tx to~~ Rx Tx MMPDU from that TA. MSDUs and MMPDUs or A-MSDUs that do not match these conditions are sent and are received ~~without~~ in the ~~benefit of encryption~~clear.

8.7.2.1 Per-MSDU Tx pseudo-code

Insert a new subclause after 8.7.2.1 as follows:

8.7.2.1a Per-MMPDU Tx pseudo-code

```
if (dot11RSNAEnabled = TRUE) then
    if ((dot11RSNAProtectedManagementFramesEnabled = TRUE) and (FrameControl.SubType
    is one of Disassociation, Deauthentication or Action)) then
        // Management Frame Protection is enabled and frame is eligible for protection
        if (MMPDU has an individual RA) then
            if (MMPDU Protection for RA is off for Tx)) then
                // Check for legacy operation
                Transmit the MMPDU without protection, after fragmentation
            else if (Pairwise key exists for the MMPDU's RA) then
                // Note that it is assumed that no entry in the key
                // mapping table will be of an unsupported cipher.
                Set the Key ID subfield of the IV field to zero
                Transmit the MMPDU, to be protected after fragmentation
                // see 8.7.2.2a
            else
                // pairwise key was not found
                Discard the entire MMPDU and generate an MLME.confirm primitive, if it
                exists, to notify the SME that the MMPDU was undeliverable
             endif
        else // MMPDU has a multicast/broadcast RA
```

```
                 if (IGTK exists) then
                     // if we find a suitable IGTK
                     Set the Key ID subfield of the MMIE to corresponding IGTK   KeyID
                     Transmit the MMPDU with BIP
                 else
                     Discard the entire MMPDU and generate an MLME.confirm primitive, if it
                     exists, to notify the SME that the MMPDU was undeliverable
                 endif
             endif
         else
             // Either Management Frame Protection is not enabled OR
             // frame is not eligible for protection
             Transmit the MMPDU without protection
         endif
     endif
```

Insert a new sub clause after 8.7.2.2:

8.7.2.2a Per-MPDU Tx pseudo-code for MMPDU

```
if ((dot11RSNAEnabled = TRUE) then
    if (MPDU is member of an MMPDU that is to be transmitted without   protection) then
        Transmit the MPDU without protection
     else if (MPDU is individual RA) then
        Protect the MPDU using entry's TK and selected cipher from RSN IE
        Transmit the MPDU
    else
        // MPDU has a multicast/broadcast RA
        Protect the MPDU using IGTK and BIP
        Transmit the MPDU
     endif
endif
```

Insert a new sub clause after 8.7.2.3 :

8.7.2.3a Per-MPDU Rx pseudo-code for an MMPDU

```
if (dot11RSNAEnabled = TRUE) then
    if (dot11RSNAProtectedManagementFramesEnabled = TRUE) then
        if (Protection for TA is off for Rx) then
            // (dot11RSNAUnprotectedManagementFramesAllowed = TRUE for Rx) and
            // TA does not support Management Frame Protection
            Receive the unencrypted MPDU
            if (Protected Frame subfield of the Frame Control field is set to 1) then
        Discard the frame
    else
        Receive the MPDU
    endif
else //Management Frame Protection is expected
    if (MPDU has individual RA) then
        if (Protected Frame subfield of the Frame Control field is set to 0) then
            //unprotected frame
            if ((Pairwise key exists) or ((Pairwise key does not exist) and (FrameControl.SubType is Action))) then
              Discard the frame
              if (security association has an AES-CCM key) then
                Increment dot11RSNAStatsCMACICVErrors
              endif
            endif
        if (security association has an AES-CCM key) then
            if (PN is not sequential) then
              Discard the MPDU as a replay
              Increment dot11RSNAStatsCCMPReplays
            else
              Decrypt frame using AES-CCM key
              if (the integrity check fails) then
                Discard the frame
                Increment dot11RSNAStatsCCMPDecryptErrors
              else
                Make the MPDU available for further processing
              endif
            endif
          else
            // if (any other cipher exists) then
            //          Process the frame using other cipher
            // else
            //          Discard the frame
            // endif
        endif
    else if (MPDU has multicast/broadcast RA) then
        if (MMIE is not present) then
            // Unprotected frame
            if (IGTK exists) or ((IGTK does not exist) and (FrameControl.SubType is Action)) then
              Discard the frame
              if (security association has an AES-128-CMAC IGTK) then
                Increment dot11RSNAStatsCMACICVError
```

```
                                  endif
                                endif
                          else if (security association has an AES-128-CMAC IGTK) then
                                if (PN is not valid) then
                                   Discard the MPDU as a replay
                                  Increment dot11RSNAStatsCMACReplays
                                else // Check integrity of the frame using AES-128-CMAC key
                                  if (the ICV fails) then
                                    Discard the frame
                                    Increment dot11RSNAStatsCMACICVErrors
                                  else
                                    Make the MPDU available for further processing
                                  endif
                                endif
                          else
                                // if (any other cipher exists) then
                                //          Process the frame using other cipher
                                // else
                                //          Discard the frame
                                // endif
                          endif
                    endif // if (MPDU has individual RA)then
              endif // if (Protection for TA is true for Rx) then
        else // if (dot11RSNAProtectedManagementFramesEnabled = TRUE) then
              Receive the MPDU
        endif
  endif
```

Insert a new sub clause after 8.7.2.4 :

8.7.2.4a Per-MMPDU Rx pseudo-code

```
  if (dot11RSNAEnabled = TRUE) then
        if (dot11RSNAProtectedManagmentFramesEnabled = TRUE) then
              if (the MPDU was not protected) then
                    Receive the MMPDU unprotected
                    Make the MMPDU available to higher layers
              else //Have a protected MMPDU
                    if ((MMPDU has individual RA) and (security association has an AES-CCM key))
                    then
                          if (the MPDU has only one MPDU or multiple MPDUs with sequential PNs)
                          then
                                Receive the MMPDU protected
                                Make the MMPDU available to higher layers
                          else
                                Discard the MMPDU as a replay
                                Increment dot11RSNAStatsRobustMgmtCCMPReplays
                          endif
                    else if ((MPDU has broadcast/multicast RA) and (security association has an AES-
                    128-CMAC IGGTK)) then
                          Receive the MMPDU
                          Make the MMPDU available to higher layers
                    else
                          // if (any other cipher exists) then
                          // Process the frame using other cipher
                          // else
                          // Discard the frame
                          // endif
                    endif
              endif
        endif
  endif
```

10. Layer Management

10.3 MLME SAP interface

10.3.17 SetKeys

10.3.17.1 MLME-SETKEYS.request

10.3.17.1.2 Semantics of the service primitive

Change the ''KeyID' and Key Type' entries in the SetKeyDescriptor of Clause 10.3.17.1.2 as follows:

Name	Type	Valid range	Description
Key ID	Integer	0-3 (or 4-4095 for IGTK)	Key identifier
Key Type	Integer	Group, Pairwise, Peer-key, IGTK	Defines whether this key is a group key, pairwise key, ~~or~~ PeerKey, or Integrity Group key.

10.3.18.1 MLME-DELETEKEYS.request

10.3.18.1.2 Semantics of the service primitive

Change the 'Protect Type' and ‘Key Type’ entries in the SetKeyDescriptor of Clause 10.3.18.1.2 as follows:

Name	**Type**	**Valid range**	**Description**
Key Type	Integer	Group, Pairwise, Peer-key, IGTK	Defines whether this key is a group key, pairwise key, ~~or~~ PeerKey, or Integrity Group key.

10.3.22.1 MLME-SETPROTECTION.request

10.3.22.1.2 Semantics of the service primitive

Change the ProtectList of Clause 10.3.22.1.2 as follows:

Name	**Type**	**Valid range**	**Description**
Protect Type	Enumeration	None, Rx, Tx, Rx_Tx, Rx_MMPDU, Tx_MMPDU, Rx_Tx_MMPDU	The protection value for this MAC.
Key Type	Integer	Group, Pairwise, Peer-key, IGTK	Defines whether this key is a group key, pairwise key, ~~or~~ PeerKey or Integrity Group key.

11. MLME

EDITORIAL NOTE: TGn has added sections up through 11.18, TGw succeeds with 11.19.

Insert at the end of Clause 11 a new sub clause as follows:

11.19 Broadcast and multicast Robust Management ~~Frame~~ frame procedures

When Management Frame Protection is enabled, the MLME shall provide an encapsulation service for broadcast/multicast Robust Management frames. All broadcast/multicast Robust Management frames shall be submitted to this service for encapsulation and transmission.

The broadcast/multicast frame protection service shall take the following actions:

— Management Frame Protection for multicast/broadcast shall be set using the MLME-SETPROTECTION.request with the Protectlist including a Key Type value of IGTK. A non-AP STA shall also set the Protect Type value to Rx_MMPDU. In an IBSS, STAs shall set the ProtectType value to Rx_Tx_MMPDU. An AP shall set the Protect Type value to Tx_MMPDU.

— The IGTK shall be installed using the MLME-SETKEYS.request with the value IGTK for the Key Type field in the Key Descriptor element.

— All broadcast/multicast Robust Management frames shall be encapsulated and protected using BIP (see 8.3.4).

11A. Fast BSS Transition

EDITORIAL NOTE: This clause is introduced by TGr and is tracked by TGw.

11A.2 Key holders

11A.2.2 Authenticator key holders

Change the 2nd dashed item in the 7th paragraph of 11A.2.2 as follows:

— The R1KH shall provide the IEEE 802.11 Authenticator function to derive and distribute the GTK and IGTK to all connected STAs.

11A.4 Fast BSS Transition Initial Mobility Domain Association

11A.4.2 Fast BSS Transision initial mobility domain association in an RSN

Change the 3rd message of the 12th paragraph in 11A.4.2 as follows:

R1KH →[#688] S1KH: Data(EAPOL-Key(1, 1, 1, 1, P, 0, ANonce, MIC,
RSNIE[PMKR1Name], MDIE, GTK[N], IGTK[M],
FTIE, TIE[ReassociationDeadline], TIE[KeyLifetime]))

11A.6 Fast BSS Transition resource request protocol

11A.6.2 Over-the-air fast BSS transition with resource request

Change the 12th paragraph in 11A.6.2 as follows:

In an RSN, on successful completion of the FT Authentication exchange of the FT resource request protocol, the PTKSA has been established and proven live. The Key Replay Counter shall be initialized to zero and the subsequent EAPOL-key frames (e.g.~~,~~ GTK and IGTK updates) shall use the Key Replay Counter to ensure they are not replayed. The PTKSA shall be deleted by the Target AP if it does not receive a reassociation request from the STA within the reassociation deadline timeout value.

11A.6.3 Over-the-DS fast BSS transition with resource request

Change the 10th paragraph in 11A.6.3 as follows:

In an RSN, on successful completion of the FT Confirm/Acknowledgement frame exchange, the PTKSA has been established and proven live. The Key Replay Counter shall be initialized to zero and the subsequent EAPOL-key frames (e.g.~~,~~ GTK and IGTK updates) shall use the Key Replay Counter to ensure they are not replayed. The PTKSA shall be deleted by the Target AP if it does not receive a reassociation request from the STA within the reassociation deadline timeout value. Resource request procedures are specified in 11A.11.

11A.7 Fast BSS transition reassociation

11A.7.1 Fast BSS transition reassociation in an RSN

Change the 2nd message of the 2nd paragraph in 11A.7.1 as follows:

Target AP →[#688] STA: Reassociation Response(RSNIE[PMKR1Name], MDIE, FTIE[MIC,
ANonce, SNonce, R1KH-ID, R0KH-ID], GTK[N],
IGTK[M]], RIC-Response)

11A.8 Fast BSS transition authentication sequence

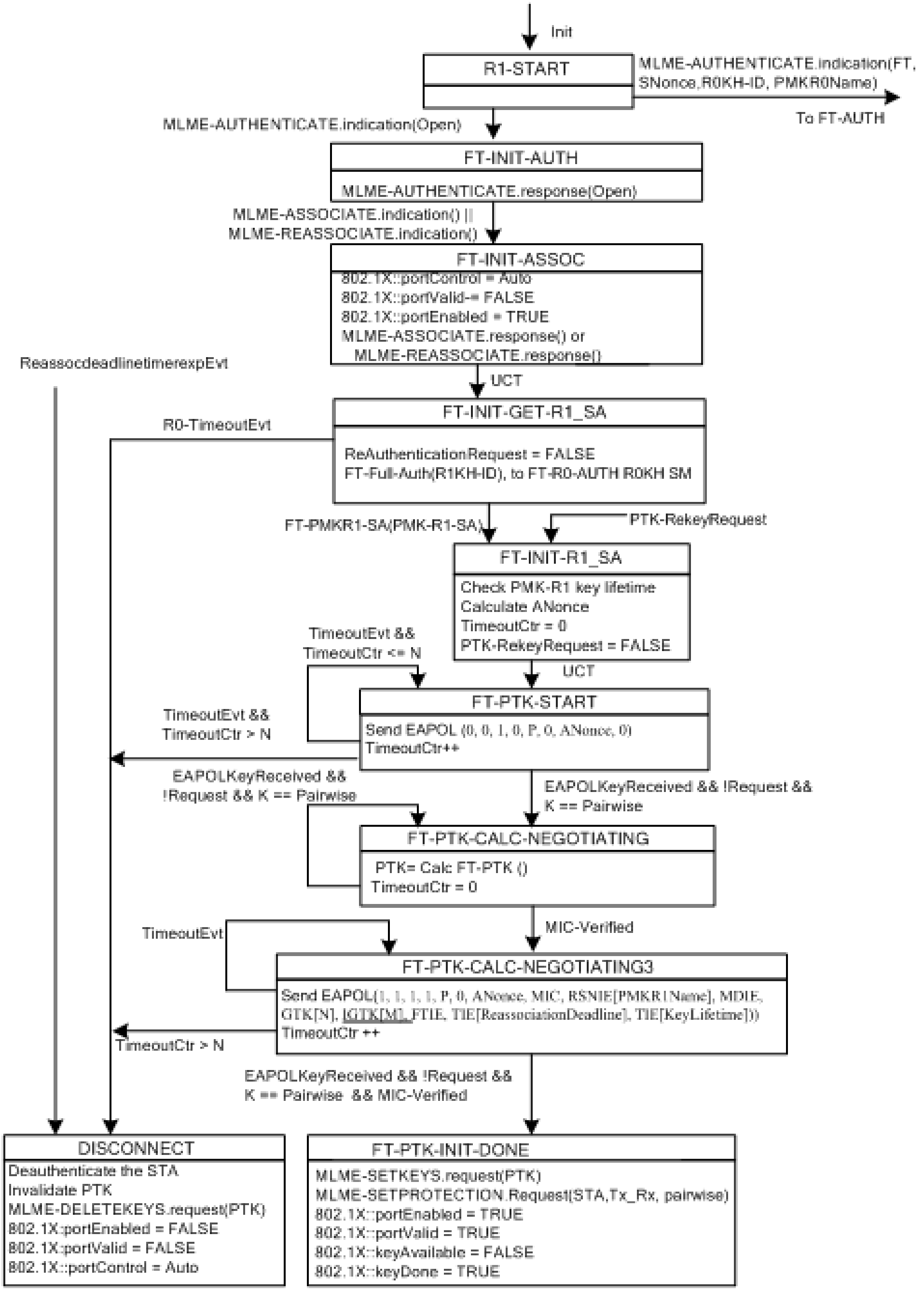
11A.8.5 FT authentication sequence: contents of fourth message

Change the 3rd dashed item of the 4th paragraph in 11A.8.5 as follows:

— When this message of the authentication sequence appears in a Reassociation Response frame, the optional parameters in the FTIE may include ~~a~~ the GTK and IGTK sub-elements. If a GTK or an IGTK are ~~is~~ included, the Key field of the sub-element shall be encrypted using KEK and the NIST AES Key Wrap algorithm. The Key field shall be padded before encrypting if the key length is less than 16 octets or if it is not a multiple of 8. The padding consists of appending a single octet 0xdd followed by zero or more 0x00 octets. When processing a received message, the receiver shall ignore this trailing padding. Addition of padding does not change the value of the Key Length field. Note: The length of the encrypted Key field can be determined from the length of the GTK or IGTK sub-element.

11A.9 Fast BSS Transition security architecture state machines

Replace Figure 11A-13 with the following Figure, with the updates being including "IGTK[M]" in the FT-PTK-CALC-NEGOTIATING3 box:


Init
R1-START
MLME-AUTHENTICATE.indication(FT, SNonce,R0KH-ID, PMKR0Name)
To FT-AUTH
MLME-AUTHENTICATE.indication(Open)
FT-INIT-AUTH
MLME-AUTHENTICATE.response(Open)
MLME-ASSOCIATE.indication() ||
MLME-REASSOCIATE.indication()
FT-INIT-ASSOC
802.1X::portControl = Auto
802.1X::portValid-= FALSE
802.1X::portEnabled = TRUE
MLME-ASSOCIATE.response() or
MLME-REASSOCIATE.response()
ReassocdeadlinetimerexpEvt
UCT
FT-INIT-GET-R1_SA
R0-TimeoutEvt
ReAuthenticationRequest = FALSE
FT-Full-Auth(R1KH-ID), to FT-R0-AUTH R0KH SM
FT-PMKR1-SA(PMK-R1-SA)
PTK-RekeyRequest
FT-INIT-R1_SA
Check PMK-R1 key lifetime
Calculate ANonce
TimeoutCtr = 0
PTK-RekeyRequest = FALSE
TimeoutEvt &&
TimeoutCtr <= N
UCT
FT-PTK-START
TimeoutEvt &&
TimeoutCtr > N
Send EAPOL (0, 0, 1, 0, P, 0, ANonce, 0)
TimeoutCtr++
EAPOLKeyReceived &&
!Request && K == Pairwise
EAPOLKeyReceived && !Request &&
K == Pairwise
FT-PTK-CALC-NEGOTIATING
PTK= Calc FT-PTK ()
TimeoutCtr = 0
TimeoutEvt
MIC-Verified
FT-PTK-CALC-NEGOTIATING3
Send EAPOL(1, 1, 1, 1, P, 0, ANonce, MIC, RSNIE[PMKR1Name], MDIE,
GTK[N], IGTK[M], FTIE, TIE[ReassociationDeadline], TIE[KeyLifetime])
TimeoutCtr ++
TimeoutCtr > N
EAPOLKeyReceived && !Request &&
K == Pairwise && MIC-Verified
DISCONNECT
Deauthenticate the STA
Invalidate PTK
MLME-DELETEKEYS.request(PTK)
802.1X:portEnabled = FALSE
802.1X:portValid = FALSE
802.1X::portControl = Auto
FT-PTK-INIT-DONE
MLME-SETKEYS.request(PTK)
MLME-SETPROTECTION.Request(STA,Tx_Rx, pairwise)
802.1X::portEnabled = TRUE
802.1X::portValid = TRUE
802.1X::keyAvailable = FALSE
802.1X::keyDone = TRUE

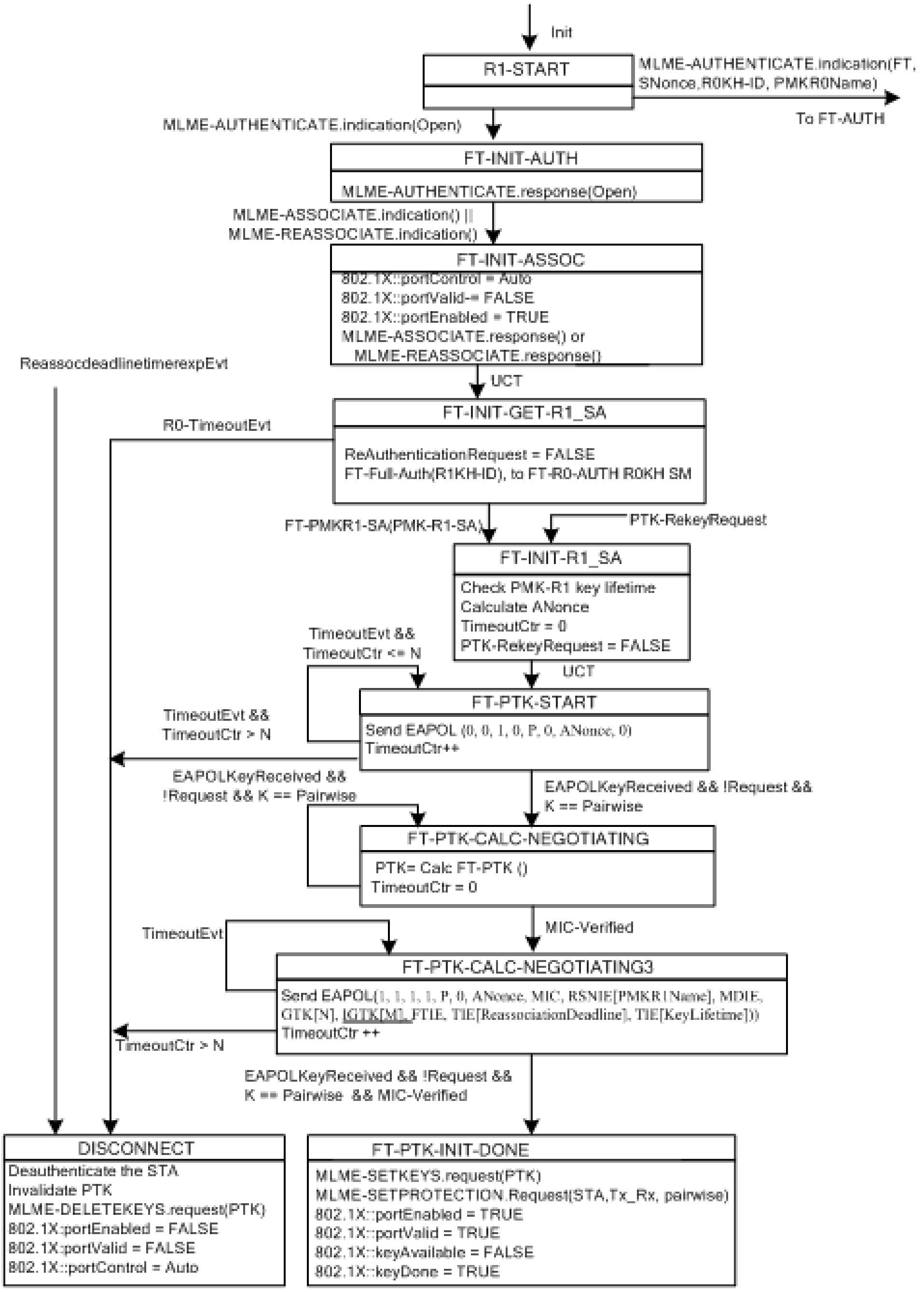
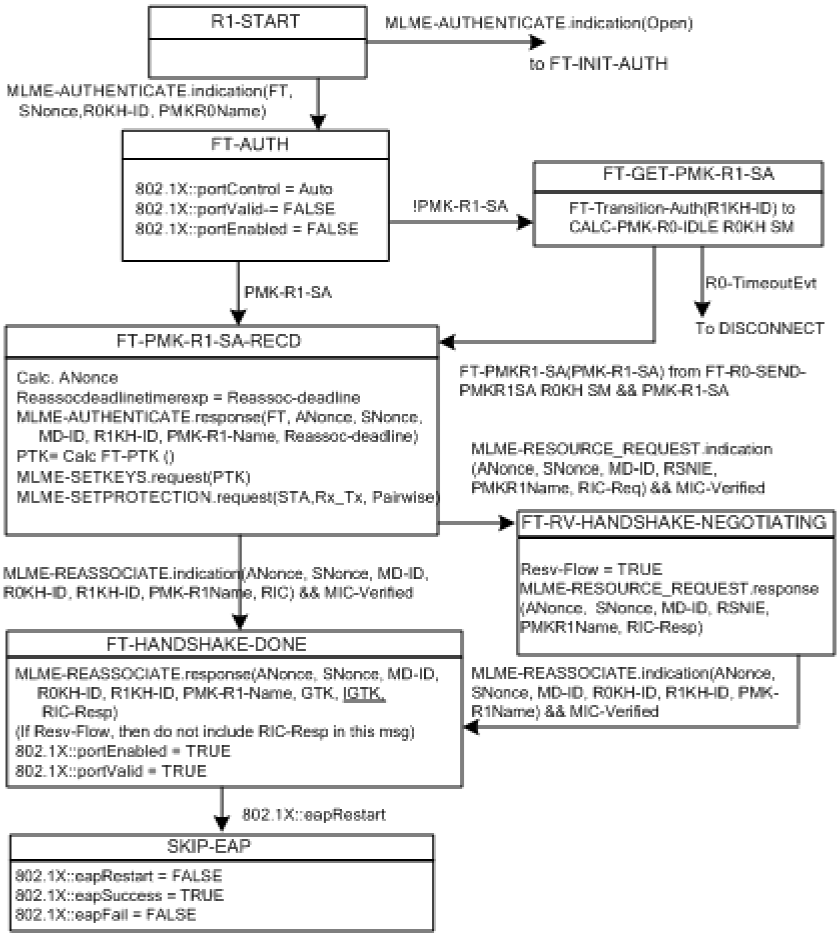

Init
R1-START
MLME-AUTHENTICATE.indication(FT, SNonce,R0KH-ID, PMKR0Name)
To FT-AUTH
MLME-AUTHENTICATE.indication(Open)
FT-INIT-AUTH
MLME-AUTHENTICATE.response(Open)
MLME-ASSOCIATE.indication() ||
MLME-REASSOCIATE.indication()
FT-INIT-ASSOC
802.1X::portControl = Auto
802.1X::portValid = FALSE
802.1X::portEnabled = TRUE
MLME-ASSOCIATE.response() or
MLME-REASSOCIATE.response()
ReassocdeadlinetimerexpEvt
UCT
FT-INIT-GET-R1_SA
R0-TimeoutEvt
ReAuthenticationRequest = FALSE
FT-Full-Auth(R1KH-ID), to FT-R0-AUTH R0KH SM
FT-PMKR1-SA(PMK-R1-SA)
PTK-RekeyRequest
FT-INIT-R1_SA
Check PMK-R1 key lifetime
Calculate ANonce
TimeoutCtr = 0
PTK-RekeyRequest = FALSE
TimeoutEvt &&
TimeoutCtr <= N
UCT
FT-PTK-START
TimeoutEvt &&
TimeoutCtr > N
Send EAPOL (0, 0, 1, 0, P, 0, ANonce, 0)
TimeoutCtr++
EAPOLKeyReceived &&
!Request && K == Pairwise
EAPOLKeyReceived && !Request &&
K == Pairwise
FT-PTK-CALC-NEGOTIATING
PTK= Calc FT-PTK ()
TimeoutCtr = 0
TimeoutEvt
MIC-Verified
FT-PTK-CALC-NEGOTIATING3
Send EAPOL(1, 1, 1, 1, P, 0, ANonce, MIC, RSNIE[PMKR1Name], MDIE,
GTK[N], IGTK[M], FTIE, TIE[ReassociationDeadline], TIE[KeyLifetime]))
TimeoutCtr ++
TimeoutCtr > N
EAPOLKeyReceived && !Request &&
K == Pairwise && MIC-Verified
DISCONNECT
Deauthenticate the STA
Invalidate PTK
MLME-DELETEKEYS.request(PTK)
802.1X:portEnabled = FALSE
802.1X:portValid = FALSE
802.1X::portControl = Auto
FT-PTK-INIT-DONE
MLME-SETKEYS.request(PTK)
MLME-SETPROTECTION.Request(STA,Tx_Rx, pairwise)
802.1X::portEnabled = TRUE
802.1X::portValid = TRUE
802.1X::keyAvailable = FALSE
802.1X::keyDone = TRUE

Figure 11A-13—Authenticator R1KH state machine (part 1)

Replace Figure 11A-14 with the following Figure, with the updates being including "IGTK" in the FT-HANDSHAKE-DONE box:


R1-START
MLME-AUTHENTICATE.indication(Open)
to FT-INIT-AUTH
MLME-AUTHENTICATE.indication(FT, SNonce,R0KH-ID, PMKR0Name)
FT-AUTH
802.1X::portControl = Auto
802.1X::portValid-= FALSE
802.1X::portEnabled = FALSE
!PMK-R1-SA
FT-GET-PMK-R1-SA
FT-Transition-Auth(R1KH-ID) to CALC-PMK-R0-IDLE R0KH SM
R0-TimeoutEvt
To DISCONNECT
PMK-R1-SA
FT-PMK-R1-SA-RECD
Calc. ANonce
Reassocdeadlinetimerexp = Reassoc-deadline
MLME-AUTHENTICATE.response(FT, ANonce, SNonce, MD-ID, R1KH-ID, PMK-R1-Name, Reassoc-deadline)
PTK= Calc FT-PTK ()
MLME-SETKEYS.request(PTK)
MLME-SETPROTECTION.request(STA,Rx_Tx, Pairwise)
FT-PMKR1-SA(PMK-R1-SA) from FT-R0-SEND-PMKR1SA R0KH SM && PMK-R1-SA
MLME-RESOURCE_REQUEST.indication (ANonce, SNonce, MD-ID, RSNIE, PMKR1Name, RIC-Req) && MIC-Verified
FT-RV-HANDSHAKE-NEGOTIATING
Resv-Flow = TRUE
MLME-RESOURCE_REQUEST.response (ANonce, SNonce, MD-ID, RSNIE, PMKR1Name, RIC-Resp)
MLME-REASSOCIATE.indication(ANonce, SNonce, MD-ID, R0KH-ID, R1KH-ID, PMK-R1Name, RIC) && MIC-Verified
FT-HANDSHAKE-DONE
MLME-REASSOCIATE.response(ANonce, SNonce, MD-ID, R0KH-ID, R1KH-ID, PMK-R1-Name, GTK, IGTK, RIC-Resp)
(If Resv-Flow, then do not include RIC-Resp in this msg)
802.1X::portEnabled = TRUE
802.1X::portValid = TRUE
MLME-REASSOCIATE.indication(ANonce, SNonce, MD-ID, R0KH-ID, R1KH-ID, PMK-R1Name) && MIC-Verified
802.1X::eapRestart
SKIP-EAP
802.1X::eapRestart = FALSE
802.1X::eapSuccess = TRUE
802.1X::eapFail = FALSE

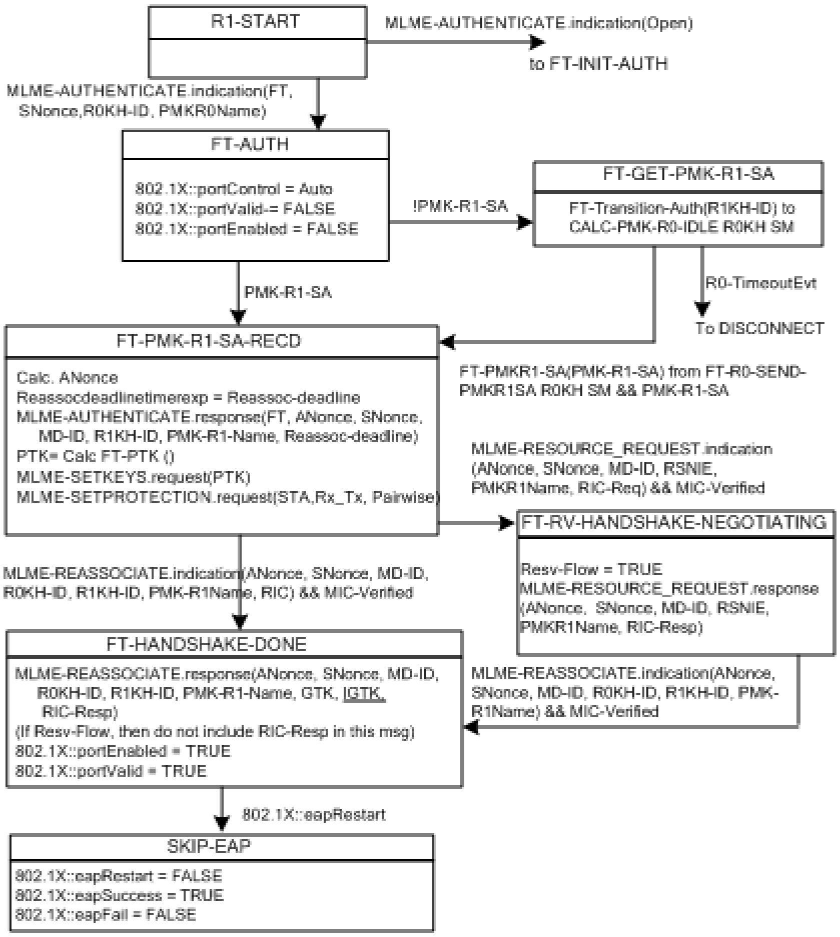

R1-START
MLME-AUTHENTICATE.indication(Open)
to FT-INIT-AUTH
MLME-AUTHENTICATE.indication(FT, SNonce,R0KH-ID, PMKR0Name)
FT-AUTH
802.1X::portControl = Auto
802.1X::portValid-= FALSE
802.1X::portEnabled = FALSE
!PMK-R1-SA
FT-GET-PMK-R1-SA
FT-Transition-Auth(R1KH-ID) to CALC-PMK-R0-IDLE R0KH SM
R0-TimeoutEvt
To DISCONNECT
PMK-R1-SA
FT-PMK-R1-SA-RECD
Calc. ANonce
Reassocdeadlinetimerexp = Reassoc-deadline
MLME-AUTHENTICATE.response(FT, ANonce, SNonce, MD-ID, R1KH-ID, PMK-R1-Name, Reassoc-deadline)
PTK= Calc FT-PTK ()
MLME-SETKEYS.request(PTK)
MLME-SETPROTECTION.request(STA,Rx_Tx, Pairwise)
FT-PMKR1-SA(PMK-R1-SA) from FT-R0-SEND-PMKR1SA R0KH SM && PMK-R1-SA
MLME-RESOURCE_REQUEST.indication (ANonce, SNonce, MD-ID, RSNIE, PMKR1Name, RIC-Req) && MIC-Verified
FT-RV-HANDSHAKE-NEGOTIATING
Resv-Flow = TRUE
MLME-RESOURCE_REQUEST.response (ANonce, SNonce, MD-ID, RSNIE, PMKR1Name, RIC-Resp)
MLME-REASSOCIATE.indication(ANonce, SNonce, MD-ID, R0KH-ID, R1KH-ID, PMK-R1Name, RIC) && MIC-Verified
FT-HANDSHAKE-DONE
MLME-REASSOCIATE.response(ANonce, SNonce, MD-ID, R0KH-ID, R1KH-ID, PMK-R1-Name, GTK, IGTK, RIC-Resp)
(If Resv-Flow, then do not include RIC-Resp in this msg)
802.1X::portEnabled = TRUE
802.1X::portValid = TRUE
MLME-REASSOCIATE.indication(ANonce, SNonce, MD-ID, R0KH-ID, R1KH-ID, PMK-R1Name) && MIC-Verified
802.1X::eapRestart
SKIP-EAP
802.1X::eapRestart = FALSE
802.1X::eapSuccess = TRUE
802.1X::eapFail = FALSE

Figure 11A-14—Authenticator R1KH state machine (part 2)

Annex A

(normative)

Protocol Implementation Conformance Statement (PICS)

A.4 PICS proforma — IEEE Std 802.11-2007

A.4.4 MAC protocol

A.4.4.1 MAC protocol capabilities

Change row entry "PC34" of the table in A.4.4.1 as follows:

PC34	Robust security network association (RSNA)	7.2.2, 7.3.1.4, 5.4.3.3, 8.7.2.1, 8.7.2.2, 8.7.2.3, 8.7.2.4, 11.3.1, 11.3.2, 8.3.3	O	Yes No

Insert the following row to end of table in A.4.4.1 as a subset of the RSN :

PC 34.1.10	Management Frame Protection	7.3.1.11, 7.4.2, 7.1.3.1.9, 7.3.2.25.3, 8.3.2.1.1, 8.3.2.1.2, 8.3.2.2, 8.3.2.3.4, 8.3.3.3.2, 8.3.3.3.5, 8.3.3.4.1, 8.3.3.4.3, 8.4.3, 8.7.2.1A, 8.7.2.3A, 8.7.2.4A	PC34.1:O	Yes No
PC 34.1.10.1	BIP	8.3.4, 11.18	PC34.1.10:M	Yes No
PC 34.1.10.1.1	Management MIC IE	7.3.2.53	PC34.1.10.1:M	Yes No

EDITORIAL NOTE: The entry value is shown as PCX 34.1.10 but its final value is pending ANA assignment

Annex D

(normative)

ASN.1 encoding of the MAC and PHY MIB

~~*Insert the following at the end of the* Dot11StationConfigEntry *in Annex D:*~~

Change the* `Dot11StationConfigEntry` *in Annex D by inserting a comma and the new entries at the end of the existing entries (represented below by “...”) as:

```
dot11StationConfigEntry ::=
      SEQUENCE {
          ... ,
          dot11RSNAProtectedManagementFramesEnabled  TruthValue,
          dot11RSNAUnprotectedManagementFramesAllowed  TruthValue
      }
```

Insert the following after the ~~dott11RSNAStats~~ dot11RSNAStats TABLE entries in Annex D:

EDITORIAL NOTE: TGn uses up to value 63

```
--************************************************************
--* Robust Management Frame Protection MIBs
--************************************************************

dot11RSNAProtectedManagementFramesEnabled       OBJECT-TYPE
      SYNTAX TruthValue
      MAX-ACCESS read-write
      STATUS current
      DESCRIPTION
               "This variable indicates whether or not this STA
                Protects unicast Management Frames."
                 DEFAULT { TRUE }
                 ::= { dot11StationConfigEntry 64 }



dot11RSNAUnprotectedManagementFramesAllowed     OBJECT-TYPE
      SYNTAX TruthValue
      MAX-ACCESS read-write
      STATUS current
      DESCRIPTION
            "This variable indicates whether or not this STA supports
            robust RSNA STAs which do not provide Robust Management
            ~~Frames~~ frames protection.”
                DEFAULT { FALSE }
                ::= { dot11StationConfigEntry 65}
```

Insert at the end of the dot11RSNAStatsEntry Sequence the following:

EDITORIAL NOTE: IEEE 802.11-2007 uses values up to 10

```
dot11RSNAStatsCMACICVErrors                 Counter32,
dot11RSNAStatsCMACReplays                   Counter32,
dot11RSNAStatsRobustMgmtCCMPReplays         Counter32,
dot11RSNABIPMICErrors                       Counter32


dot11RSNAStatsCMACICVErrors                 OBJECT-TYPE
      SYNTAX Counter32
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION
            "The number of received MPDUs discarded by the CMAC integrity check algorithm."
                ::= { dot11RSNAStatsEntry 11 }

dot11RSNAStatsCMACReplays             OBJECT-TYPE
      SYNTAX Counter32
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION
            "The number of received MPDUs discarded by the CMAC replay errors."
                ::= { dot11RSNAStatsEntry 12 }


dot11RSNAStatsRobustMgmtCCMPReplays OBJECT-TYPE
      SYNTAX Counter32
      MAX-~~ACESS ~~ACCESS read-only
      STATUS current
~~    DESCRIPTION: ~~
      DESCRIPTION:
            "The number of received MMPDUs discarded due to CCMP replay errors"
             ::= {dot11RSNAStatsEntry 13}

dot11RSNABIPMICErrors OBJECT-TYPE
      SYNTAX Counter32
      MAX-~~ACESS ~~ACCESS read-only
      STATUS current
~~    DESCRIPTION: ~~
      DESCRIPTION:
            "The number of received MMPDUs discarded due to BIP MIC errors"
             ::= {dot11RSNAStatsEntry 14}
```

```
--**********************************************************
--* End of Robust Management Frame MIB
--**********************************************************
```

Annex H

(informative)

Insert the following at the end of Annex H:

H.8 Test vectorsfor AES-128-CMAC

“Test vectors for AES-128-CMAC may be found in Annex D.1 of NIST SP-800-38B”