IEEE P802.15
Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)	
Title	**Security Clauses – Streamlined Version, with Security Level Set**	
Date	[13-Nov-08]	
Source	René Struik Certicom Corp. 5520 Explorer Drive, 4th Floor Mississauga, ON L4W 5L1	E-mail: rstruik@certicom.com Phone: +1 (905) 501-6083 Fax: +1 (905) 507-4230
Re:	Security and Efficiency Enhancements for IEEE 802.15.4e	
Abstract	This document provides a streamlined version of Clauses 7.5.8 and 7.6 of IEEE 802.15.4-2006.	
Purpose	Security and efficiency enhancements of IEEE 802.15.4-2006	
Notice	This document has been prepared to assist the IEEE P802.15. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.	
Release	The contributor acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15.	

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress* and a direction and length corresponding to one of its GTSs, the device shall adjust the starting slot of the GTS corresponding to the GTS descriptor and start using it immediately.

In cases where it is necessary for the PAN coordinator to include a GTS descriptor in its beacon, it shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length. After *aGTSDescPersistenceTime* superframes, the PAN coordinator shall remove the GTS descriptor from the beacon.

7.5.7.6 GTS expiration

The MLME of the PAN coordinator shall attempt to detect when a device has stopped using a GTS using the following rules:

— For a transmit GTS, the MLME of the PAN coordinator shall assume that a device is no longer using its GTS if a data frame is not received from the device in the GTS at least every 2**n* superframes, where *n* is defined below.
— For receive GTSs, the MLME of the PAN coordinator shall assume that a device is no longer using its GTS if an acknowledgment frame is not received from the device at least every 2**n* superframes, where *n* is defined below. If the data frames sent in the GTS do not require acknowledgment frames, the MLME of the PAN coordinator will not be able to detect whether a device is using its receive GTS. However, the PAN coordinator is capable of deallocating the GTS at any time.

The value of *n* is defined as follows:

$n = 2^{(8-macBeaconOrder)}$ $0 \le macBeaconOrder \le 8$

$n = 1$ $9 \le macBeaconOrder \le 14$

7.5.8 Frame security

The MAC sublayer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. IEEE P802.15.4REVb/D6 supports the following security services (see 5.5.6 for definitions):

— Data confidentiality
— Data authenticity
— Replay protection

The information determining how to provide the security is found in the security-related PIB (see Table 88).

7.5.8.1 Security-related MAC PIB attributes

The security-related MAC PIB attributes contain:

— Key table (*macKeyTable*, *macKeyTableEntries*)
— Device table (*macDeviceTable*, *macDeviceTableEntries*)
— Minimum security level table (*macSecurityLevelTable*, *macSecurityLevelTableEntries*)
— Frame counter (*macFrameCounter*)
— Automatic request attributes (*macAutoRequestSecurityLevel*, *macAutoRequestKeyIdMode*, *macAutoRequestKeySource*, *macAutoRequestKeyIndex*)
— Default key source (*macDefaultKeySource*)
— PAN coordinator address (*macPANCoordExtendedAddress, macPANCoordShortAddress*)
— Key source table (*macKeySourceTable, macKeySourceTableEntries*)

7.5.8.1.1 Key table

The key table holds key descriptors (keys with related key-specific information) that are required for security processing of outgoing and incoming frames. Key-specific information in the key table is identified based on information explicitly contained in the requesting primitive or in the received frame, as described in the outgoing frame key retrieval procedure (see 7.5.8.2.2) and the incoming frame key retrieval procedure (see 7.5.8.2.4), as well as in the KeyDescriptor lookup procedure (see 7.5.8.2.6) and the KeySourceDescriptor lookup procedure (see 7.5.8.2.9).

7.5.8.1.2 Device table

The device table holds device descriptors (device-specific addressing information and security-related information) that, when combined with key-specific information from the key table, provide all the keying material needed to secure outgoing (see 7.5.8.2.1) and unsecure incoming frames (see 7.5.8.2.3). Device-specific information in the device table is identified based on the originator of the frame, as described in the incoming frame device retrieval procedure (see 7.5.8.2.5) and the DeviceDescriptor lookup procedure (see 7.5.8.2.8).

7.5.8.1.3 Minimum security level table

The minimum security level table holds information regarding the security level the device expects to have been applied by the originator of a frame, depending on frame type and, if it concerns a MAC command frame, the command frame identifier. Security processing of an incoming frame will fail if the frame is not adequately protected, as described in the incoming frame security procedure (see 7.5.8.2.3) and in the incoming security level checking procedure (see 7.5.8.2.11).

7.5.8.1.4 Frame counter

The 4-octet frame counter is used to provide replay protection and semantic security of the cryptographic building block used for securing outgoing frames. The frame counter is included in each secured frame and is one of the elements required for the unsecuring operation at the recipient(s). The frame counter is incremented each time an outgoing frame is secured, as described in the outgoing frame security procedure (see 7.5.8.2.1). When the frame counter reaches its maximum value of 0xffffffff, the associated keying material can no longer be used, thus requiring all keys associated with the device to be updated. This provides a mechanism for ensuring that the keying material for every frame is unique and, thereby, provides for sequential freshness.

7.5.8.1.5 Automatic request attributes

Automatic request attributes hold all the information needed to secure outgoing frames generated automatically and not as a result of a higher layer primitive, as is the case with automatic data requests.

7.5.8.1.6 Default key source

The default key source is information commonly shared between originator and receipient(s) of a secured frame, which, when combined with additional information explicitly contained in the requesting primitive or in the received frame, allows an originator or a recipient to determine the key required for securing or unsecuring this frame, respectively. This provides a mechanism for significantly reducing the overhead of security information contained in secured frames in particular use cases (see 7.5.8.2.2 and 7.5.8.2.4).

7.5.8.1.7 PAN coordinator address

The address of the PAN coordinator is information commonly shared between all devices in a PAN, which, when combined with additional information explicitly contained in the requesting primitive or in the

received frame, allows an originator of a frame directed to the PAN coordinator or a recipient of a frame originating from the PAN coordinator to determine the key and security-related information required for securing or unsecuring, respectively, this frame (see 7.5.8.2.2 and 7.5.8.2.4).

7.5.8.1.8 Key source table

The key source table holds key source descriptors (key-specific information) that, when combined with additional information explicitly contained in the requesting primitive or in the received frame, allow an originator or a recipient to determine the key required for securing or unsecuring this frame, respectively (see 7.5.8.2.2 and 7.5.8.2.4). For received frames, this information may be either implicitly derived from the addressing fields of the frame or explicitly indicated in the frame by its originator, as described in the outgoing frame key retrieval procedure (see 7.5.8.2.2) and the incoming frame key retrieval procedure (see 7.5.8.2.4).

7.5.8.2 Functional description

A device may optionally implement security. A device that does not implement security shall not provide a mechanism for the MAC sublayer to perform any cryptographic transformation on incoming and outgoing frames, nor require any PIB attributes associated with security. A device that implements security shall provide a mechanism for the MAC sublayer to provide cryptographic transformations on incoming and outgoing frames using information in the PIB attributes associated with security when the *macSecurityEnabled* attribute is set to TRUE.

If the MAC sublayer is required to transmit a frame or receives an incoming frame, the MAC sublayer shall process the frame as specified in 7.5.8.2.1 and 7.5.8.2.3, respectively.

7.5.8.2.1 Outgoing frame security procedure

The inputs to this procedure are the frame to be secured, and the SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters from the originating primitive or automatic request PIB attributes. The outputs from this procedure are the status of the procedure and, if this status is SUCCESS, the secured frame.

The outgoing frame security procedure involves the following steps:

a) If the security enabled subfield of the frame control field of the frame to be secured is set to zero, the procedure shall set the security level to zero.
b) If the security enabled subfield of the frame control field of the frame to be secured is set to one, the procedure shall set the security level to the SecurityLevel parameter. If the resulting security level is zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.
c) If the *macSecurityEnabled* attribute is set to FALSE and the security level is unequal to zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.
d) The procedure shall determine whether the frame to be secured satisfies the constraint on the maximum length of MAC frames, as follows:
    i) The procedure shall set the length M, in octets, of the authentication field to zero if the security level is equal to zero and shall determine this value from the security level and Table 96 otherwise.
    ii) The procedure shall determine the length AuxLen, in octets, of the auxiliary security header (see 7.6.2) using KeyIdMode and the security level.
    iii) The procedure shall determine the data expansion as AuxLen+M.
    iv) The procedure shall check whether the length of the frame to be secured, including data expansion and frame check sequence, is less than or equal to *aMaxPHYPacketSize*. If this check fails, the procedure shall return with a status of FRAME_TOO_LONG.
e) If the security level is zero, the procedure shall set the secured frame to be the frame to be secured and return with the secured frame and a status of SUCCESS.

f) The procedure shall set the frame counter to the *macFrameCounter* attribute. If the frame counter has the value 0xffffffff, the procedure shall return with a status of COUNTER_ERROR.

g) The procedure shall obtain the KeyDescriptor using the outgoing frame key retrieval procedure as described in 7.5.8.2.2. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

h) If the Blacklisted element of the KeyDescriptor is set to TRUE, the procedure shall return with a status of KEY_ERROR.

i) The procedure shall insert the auxiliary security header into the frame, with fields set as follows:
    i) The security level subfield of the security control field shall be set to the security level.
    ii) The key identifier mode subfield of the security control field shall be set to the KeyIdMode parameter.
    iii) The frame counter field shall be set to the frame counter.
    iv) If the KeyIdMode parameter is set to a value not equal to zero, the key source and key index subfields of the key identifier field shall be set to the KeySource and KeyIndex parameters, respectively.

j) The procedure shall then use *aExtendedAddress*, the frame counter, the security level and the Key element of the KeyDescriptor to produce the secured frame according to the CCM* transformation process described in the security operations (see 7.6.3.4).
    i) If the SecurityLevel parameter specifies the use of encryption (see Table 96), the encryption operation shall be applied only to the actual payload field within the MAC payload, i.e., the beacon payload field (see 7.2.2.1.8), command payload field (see 7.2.2.4.3), or data payload field (see 7.2.2.2.2), depending on the frame type. The corresponding payload field is passed to the CCM* transformation process described in 7.6.3.4 as the unsecured payload (see Table 98). The resulting encrypted payload shall substitute the original payload.
    ii) The remaining fields in the MAC payload part of the frame shall be passed to the CCM* transformation process described in 7.6.3.4 as the non-payload fields (see Table 98).
    iii) The ordering and exact manner of performing the encryption and integrity operations and the placement of the resulting encrypted data or integrity code within the MAC payload field shall be as defined in 7.6.3.4.

k) The procedure shall increment the frame counter by one and set the *macFrameCounter* attribute to the resulting value.

l) If the *macFrameCounter* element is equal to 0xffffffff, the procedure shall set the Blacklisted element of the KeyDescriptor to TRUE.

m) The procedure shall return with the secured frame and a status of SUCCESS.

7.5.8.2.2 Outgoing frame key retrieval procedure

The inputs to this procedure are the frame to be secured, and the KeyIdMode, KeySource and KeyIndex parameters from the originating primitive. The outputs from this procedure are a passed or failed status and, if passed, a KeyDescriptor.

The outgoing frame key retrieval procedure involves the following steps:

a) If the KeyIdMode parameter is set to 0x00 (implicit key identification), the procedure shall determine the key source lookup data and key source lookup size as follows:
    i) If the destination addressing mode subfield of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a value in the range 0x0000-0xfffd (i.e., the short address is used), the key source lookup data shall be set to the 2-octet source PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet *macPANCoordShortAddress* attribute. The key source lookup size shall be set to four.
    ii) If the destination addressing mode subfield of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to 0xfffe (i.e., the extended

address is used), the key source lookup data shall be set to the 8-octet *macPANCoordExtendedAddress* attribute. The key source lookup size shall be set to eight.

iii) If the destination addressing mode subfield of the frame control field of the frame is set to 0x02, the key source lookup data shall be set to the 2-octet destination PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet destination address field of the frame. The key source lookup size shall be set to four.

iv) If the destination addressing mode subfield of the frame control field of the frame is set to 0x03, the key source lookup data shall be set to the 8-octet destination address field of the frame. The key source lookup size shall be set to eight.

The key index shall be set to the single octet 0x00.

b) If the KeyIdMode parameter is set to a value not equal to 0x00 (explicit key identification), the procedure shall determine the key source lookup data and key source lookup size as follows:

i) If the KeyIdMode parameter is set to 0x01, the key source lookup data shall be set to the 8-octet *macDefaultKeySource* attribute. The key source lookup size shall be set to eight.

ii) If the KeyIdMode parameter is set to 0x02, the key source lookup data shall be set to the 4-octet KeySource parameter. The key source lookup size shall be set to four.

iii) If the KeyIdMode parameter is set to 0x03, the key source lookup data shall be set to the 8-octet KeySource parameter. The key source lookup size shall be set to nine.

The key index shall be set to the 1-octet KeyIndex parameter.

c) The procedure shall obtain the KeySourceDescriptor by passing the key source lookup data and the key source lookup size to the KeySourceDescriptor lookup procedure as described in 7.5.8.2.9. If that procedure returns with a failed status, the procedure shall also return with a failed status.

d) The procedure shall obtain the KeyDescriptor by passing the KeySourceDescriptor and the key index to the KeyDescriptor lookup procedure as described in 7.5.8.2.6. If that procedure returns with a failed status, this procedure shall also return with a failed status.

e) The procedure shall return with a passed status, having obtained the KeyDescriptor.

NOTE—For broadcast frames, the outgoing frame key retrieval procedure will result in a failed status if implicit key identification is used. Hence, one needs to use explicit key identification for broadcast frames.

7.5.8.2.3 Incoming frame security procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are the unsecured frame, the security level, the key identifier mode, the key source, the key index and the status of the procedure. All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure. It is assumed that the PIB attributes associating KeyDescriptors in *macKeyTable* with a single, unique device or a number of devices will have been established by the next higher layer.

The incoming frame security procedure involves the following steps:

a) If the security enabled subfield of the frame control field of the frame to be unsecured is set to zero, the procedure shall set the security level to zero.

b) If the security enabled subfield of the frame control field of the frame to be unsecured is set to one and the frame version number of the frame control field of the frame to be unsecured is set to zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNSUPPORTED_LEGACY.

c) If the security enabled subfield of the frame control field of the frame to be unsecured is set to one, the procedure shall set the security level and the key identifier mode to the corresponding subfields of the security control field of the auxiliary security header of the frame to be unsecured, and the key

source and key index to the corresponding subfields of the key identifier field of the auxiliary security header of the frame to be unsecured, if present. If the resulting security level is zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNSUPPORTED_SECURITY.

d) If the *macSecurityEnabled* attribute is set to FALSE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS if the security level is equal to zero and with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNSUPPORTED_SECURITY otherwise.

e) The procedure shall obtain the SecurityLevelDescriptor by passing the frame type and depending on whether the frame is a MAC command frame, the first octet of the MAC payload (i.e., command frame identifier for a MAC command frame) to the SecurityLevelDescriptor lookup procedure described in 7.5.8.2.10. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNAVAILABLE_SECURITY_LEVEL.

f) The procedure shall determine whether the frame to be unsecured conforms to the security level policy by passing the SecurityLevelDescriptor and the security level to the incoming security level checking procedure as described in 7.5.8.2.11. If that procedure returns with a failed status, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of IMPROPER_SECURITY_LEVEL; otherwise, if that procedure returns with a passed status and the security level is equal to zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS.

g) The procedure shall obtain the DeviceDescriptor using the incoming frame device retrieval procedure described in 7.5.8.2.5. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNAVAILABLE_DEVICE.

h) If the incoming security level checking procedure of Step f above had as output the ‘conditionally passed’ status, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS if the Exempt element of the DeviceDescriptor is set to TRUE and with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of IMPROPER_SECURITY_LEVEL otherwise.

i) The procedure shall set the frame counter to the frame counter field of the auxiliary security header of the frame to be unsecured. If the frame counter has the value 0xffffffff, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of COUNTER_ERROR.

j) The procedure shall determine whether the frame counter is greater than or equal to the FrameCounter element of the DeviceDescriptor. If this check fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of COUNTER_ERROR.

k) The procedure shall obtain the KeyDescriptor using the incoming frame key retrieval procedure described in 7.5.8.2.4. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNAVAILABLE_KEY.

l) The procedure shall obtain the KeyDeviceDescriptor using the KeyDeviceDescriptor lookup procedure described in 7.5.8.2.7. If that procedure fails or if the Blacklisted element of the KeyDeviceDecriptor is set to TRUE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of KEY_ERROR.

m) The procedure shall determine whether the frame to be unsecured conforms to the key usage policy by passing the KeyDescriptor, the frame type and depending on whether the frame is a MAC com-

mand frame, the first octet of the MAC payload (i.e., command frame identifier for a MAC command frame) to the incoming key usage policy checking procedure as described in 7.5.8.2.12. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of IMPROPER_KEY_TYPE.

n) The procedure shall then use the ExtAddress element of the DeviceDescriptor, the frame counter, the security level and the Key element of the KeyDescriptor to produce the unsecured frame according to the CCM* inverse transformation process described in the security operations (see 7.6.3.5).
    i) If the security level specifies the use of encryption (see Table 96), the decryption operation shall be applied only to the actual payload field within the MAC payload, i.e., the beacon payload field (see 7.2.2.1.8), command payload field (see 7.2.2.4.3), or data payload field (see 7.2.2.2.2), depending on the frame type. The corresponding payload field shall be passed to the CCM* inverse transformation process described in 7.6.3.5 as the secure payload.
    ii) The remaining fields in the MAC payload part of the frame shall be passed to the CCM* inverse transformation process described in 7.6.3.5 as the non-payload fields (see Table 100).
    iii) The ordering and exact manner of performing the decryption and integrity checking operations and the placement of the resulting decrypted data within the MAC payload field shall be as defined in 7.6.3.5.
o) If the CCM* inverse transformation process fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SECURITY_ERROR.
p) The procedure shall increment the frame counter by one and set the FrameCounter element of the DeviceDescriptor to the resulting value.
q) If the FrameCounter element is equal to 0xffffffff, the procedure shall set the Blacklisted element of the KeyDeviceDescriptor to TRUE.
r) The procedure shall return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS.

7.5.8.2.4 Incoming frame key retrieval procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are a passed or failed status and, if passed, a KeyDescriptor.

The incoming frame key retrieval procedure involves the following steps:

a) If the key identifier mode subfield of the security control field of the auxiliary security header of the frame is set to 0x00 (implicit key identification), the procedure shall determine the key source lookup data and the key source lookup size as follows:
    i) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a value in the range 0x0000-0xfffd (i.e., the short address is used), the key source lookup data shall be set to the 2-octet destination PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet *macPANCoordShortAddress* attribute. The key source lookup size shall be set to four.
    ii) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to 0xfffe (i.e., the extended address is used), the key source lookup data shall be set to the 8-octet *macPANCoordExtendedAddress* attribute. The key source lookup size shall be set to eight.
    iii) If the source address mode of the frame control field of the frame is set to 0x02, the key source lookup data shall be set to the 2-octet source PAN ID field of the frame, or to the 2-octet destination PAN ID field of the frame if the PAN ID compression subfield of the frame control field of the frame is set to one, right-concatenated (see B.1.1) with the 2-octet source address field of the frame. The key source lookup size shall be set to four.

iv) If the source address mode of the frame control field of the frame is set to 0x03, the key source lookup data shall be set to the 8-octet source address field of the frame. The key source lookup size shall be set to eight.

The key index shall be set to the single octet 0x00.

b) If the key identifier mode subfield of the security control field of the auxiliary security header of the frame is set to a value unequal to 0x00 (explicit key identification), the procedure shall determine the key source lookup data and key source lookup size as follows:

i) If the key identifier mode is set to 0x01, the key source lookup data shall be set to the 8-octet *macDefaultKeySource* attribute. The key source lookup size shall be set to eight.

ii) If the key identifier mode is set to 0x02, the key source lookup data shall be set to the 4-octet key source subfield of the key identifier field of the auxiliary security header. The key source lookup size shall be set to four.

iii) If the key identifier mode is set to 0x03, the key source lookup data shall be set to the 8-octet key source subfield of the key identifier field of the auxiliary security header. The key source lookup size shall be set to eight.

The key index shall be set to the 1-octet key index subfield of the key identifier field of the auxiliary security header.

c) The procedure shall obtain the KeySourceDescriptor by passing the key source lookup data and the key source lookup size to the KeySourceDescriptor lookup procedure as described in 7.5.8.2.9. If that procedure returns with a failed status, the procedure shall also return with a failed status.

d) The procedure shall obtain the KeyDescriptor by passing the KeySourceDescriptor and key index to the KeyDescriptor lookup procedure as described in 7.5.8.2.6. If that procedure returns with a failed status, the procedure shall also return with a failed status.

e) The procedure shall return with a passed status having obtained the KeyDescriptor.

7.5.8.2.5 Incoming frame device retrieval procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are a passed or failed status and, if passed, a DeviceDescriptor.

The incoming frame device retrieval procedure involves the following steps:

a) The procedure shall determine the device lookup data and the device lookup size as follows:

i) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a value in the range 0x0000-0xfffd (i.e., the short address is used), the device lookup data shall be set to the 2-octet destination PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet *macPANCoordShort-Address* attribute. The device lookup size shall be set to four.

ii) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a 0xfffe (i.e., the extended address is used), the device lookup data shall be set to the 8-octet *macPANCoordExtendedAddress* attribute. The device lookup size shall be set to eight.

iii) If the source address mode of the frame control field of the frame is set to 0x02, the device lookup data shall be set to the 2-octet source PAN ID field of the frame, or to the 2-octet destination PAN ID field of the frame if the PAN ID compression subfield of the frame control field of the frame is set to one, right-concatenated (see B.1.1) with the 2-octet source address field of the frame. The device lookup size shall be set to four.

iv) If the source address mode of the frame control field of the frame is set to 0x03, the device lookup data shall be set to the 8-octet source address field of the frame. The device lookup size shall be set to eight.

b) The procedure shall obtain the DeviceDescriptor by passing the device lookup data and the device lookup size to the DeviceDescriptor lookup procedure as described in 7.5.8.2.8. If that procedure returns with a failed status, the procedure shall also return with a failed status.
c) The procedure shall return with a passed status having obtained the DeviceDescriptor.

7.5.8.2.6 KeyDescriptor lookup procedure

The inputs to this procedure are the KeySourceDescriptor and the key index. The outputs from this procedure are a passed or failed status and, if passed, a KeyDescriptor.

The KeyDescriptor lookup procedure involves the following steps:

a) For each KeyDescriptor in the *macKeyTable* attribute, the procedure shall check whether the ExtKeySource element of the KeyDescriptor is equal to the corresponding element of the KeySourceDescriptor and whether the KeyIndex element of the KeyDescriptor is equal to the key index parameter. If both checks pass (i.e., there is a match), the procedure shall return with this (matching) KeyDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.7 KeyDeviceDescriptor lookup procedure

The inputs to this procedure are the KeyDescriptor and the DeviceDescriptor. The outputs from this procedure are a passed or failed status and, if passed, a KeyDeviceDescriptor.

The KeyDeviceDescriptor lookup procedure involves the following steps:

a) For each KeyDeviceDescriptor in the KeyDeviceList of the KeyDescriptor, the procedure shall check whether the ExtAddress element of the DeviceDescriptor is equal to the DeviceAddress element of the KeyDeviceDescriptor. If this check passes (i.e., there is a match), the procedure shall return with the KeyDeviceDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.8 DeviceDescriptor lookup procedure

The inputs to this procedure are the device lookup data and the device lookup size. The output from this procedure are a passed or failed status and, if passed, a DeviceDescriptor.

The DeviceDescriptor lookup procedure involves the following steps:

a) For each DeviceDescriptor in the *macDeviceTable* attribute:
    i) If the device lookup size is four and the device lookup data is equal to the PAN ID element of the DeviceDescriptor right-concatenated (see B.1.1) with the ShortAddress element of the DeviceDescriptor (i.e., there is a match), this procedure shall return with the DeviceDescriptor and a passed status.
    ii) If the device lookup size is eight and the device lookup data is equal to the ExtAddress element of the DeviceDescriptor (i.e., there is a match), this procedure shall return with the DeviceDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.9 KeySourceDescriptor lookup procedure

The inputs to this procedure are the key source lookup data and the key source lookup size. The output from this procedure are a passed or failed status and, if passed, a KeySourceDescriptor.

The KeySourceDescriptor lookup procedure involves the following steps:

a) For each KeySourceDescriptor in the *macKeySourceTable* attribute:
   i) If the key source lookup size is four and the key source lookup data is equal to the ShortKeySource element of the KeySourceDescriptor (i.e., there is a match), this procedure shall return with the KeySourceDescriptor and a passed status.
   ii) If the key source lookup size is eight and the key source lookup data is equal to the ExtKeySource element of the KeySourceDescriptor (i.e., there is a match), this procedure shall return with the KeySourceDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.10 SecurityLevelDescriptor lookup procedure

The inputs to this procedure are the frame type and the command frame identifier. The output from this procedure are a passed or failed status and, if passed, a SecurityLevelDescriptor.

The SecurityLevelDescriptor lookup procedure involves the following steps:

a) For each SecurityLevelDescriptor in the *macSecurityLevelTable* attribute:
   i) If the frame type is not equal to 0x03 and the frame type is equal to the FrameType element of the SecurityLevelDescriptor (i.e., there is a match), the procedure shall return with the SecurityLevelDescriptor and a passed status.
   ii) If the frame type is equal to 0x03, the frame type is equal to the FrameType element of the SecurityLevelDescriptor and the command frame identifier is equal to the CommandFrameIdentifier element of the SecurityLevelDescriptor, the procedure shall reutrn with the SecurityLevelDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.11 Incoming security level checking procedure

The inputs to this procedure are the SecurityLevelDescriptor and the incoming security level. The output from this procedure is a passed, failed, or ‘conditionally passed’ status.

The incoming security level checking procedure involves the following steps:

a) For each SecurityModeDescriptor in the SecurityLevelList of the SecurityLevelDescriptor, the procedure shall check whether the incoming security level is equal to the SecurityLevel element of the SecurityModeDescriptor. If this check is successful (i.e., there is a match), the procedure shall return with a passed status.
b) If the incoming security level is equal to 0x00 and the DeviceOverrideSecurityMinimum element of the SecurityLevelDescriptor is set to TRUE, the procedure shall return with a ‘conditionally passed’ status.
c) The procedure shall return with a failed status.

7.5.8.2.12 Incoming key usage policy checking procedure

The inputs to this procedure are the KeyDescriptor, the frame type and the command frame identifier. The output from this procedure is a passed or failed status.

The incoming key usage policy checking procedure involves the following steps:

a) For each KeyUsageDescriptor in the KeyUsageList of the KeyDescriptor:
   i) If the frame type is not equal to 0x03 and the frame type is equal to the FrameType element of the KeyUsageDescriptor, the procedure shall return with a passed status.

ii) If the frame type is equal to 0x03, the frame type is equal to the FrameType element of the KeyUsageDescriptor and the command frame identifier is equal to the CommandFrameIdentifier element of the KeyUsageDescriptor, the procedure shall return with a passed status.

b) The procedure shall return with a failed status.

7.6 Security suite specifications

7.6.1 PIB security material

The PIB security-related attributes are presented in Table 88, Table 89, Table 90, Table 91, Table 92, Table 93, Table 94, and Table 95.

Table 88— Security-related MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
macKeyTable	0x71	List of KeyDescriptor entries (see Table 89)	–	A table of KeyDescriptor entries, each containing keys and related information required for secured communications.	(empty)
macKeyTableEntries	0x72	Integer	Implementation specific	The number of entries in *macKeyTable*.	0
macDeviceTable	0x73	List of DeviceDescriptor entries (see Table 93)	–	A table of DeviceDescriptor entries, each indicating a remote device with which this device securely communicates.	(empty)
macDeviceTableEntries	0x74	Integer	Implementation specific	The number of entries in *macDeviceTable*.	0
macSecurityLevelTable	0x75	Table of SecurityLevelDescriptor entries (see Table 92)	–	A table of SecurityLevelDescriptor entries, each with information as to the minimum security level expected depending on incoming frame type and subtype.	(empty)
macSecurityLevelTableEntries	0x76	Integer	Implementation specific	The number of entries in *macSecurityLevelTable*.	0
macFrameCounter	0x77	Integer	0x00000000 – 0xffffffff	The outgoing frame counter for this device.	0x00000000
macAutoRequestSecurityLevel	0x78	Integer	0x00–0x07	The security level used for automatic data requests.	0x06
macAutoRequestKeyIdMode	0x79	Integer	0x00–0x03	The key identifier mode used for automatic data requests. This attribute is invalid if the *macAutoRequestSecurityLevel* attribute is set to 0x00.	0x00

Table 88— Security-related MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macAutoRequestKeySource	0x7a	As specified by the *macAutoRequestKeyIdMode* parameter	–	The originator of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* element is invalid or set to 0x00.	All octets 0xff
macAutoRequestKeyIndex	0x7b	Integer	0x01-0xff	The index of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* attribute is invalid or set to 0x00.	All octets 0xff
macDefaultKeySource	0x7c	Set of 8 octets	–	The identifier of the default key source used for key identifier mode 0x01.	All octets 0xff
macPANCoordExtendedAddress	0x7d	IEEE address	An extended 64-bit IEEE address	The 64-bit address of the PAN coordinator.	—
macPANCoordShortAddress	0x7e	Integer	0x0000–0xffff	The 16-bit short address assigned to the PAN coordinator. A value of 0xfffe indicates that the PAN coordinator is only using its 64-bit extended address. A value of 0xffff indicates that this value is unknown.	0x0000
macKeySourceTable	0x7f	List of KeySourceDescriptor entries (see Table 94)	–	A table of KeySourceDescriptor entries, each indicating key identifying information required for secured communications.	(empty)
macKeySourceTableEntries	0x80	Integer	Implementation specific	The number of entries in *macKeySourceTable*.	0

Table 89—Elements of KeyDescriptor

Name	Type	Range	Description
ExtKeySource	Set of 8 octets	–	The 64-bit identifier of the key source (see 7.6.2.4.1)
KeyIndex	Integer	0x00-0xff	The identifier of the key index (see 7.6.2.4.2). A value of 0x00 indicates an implicitly identified key; a value in the range 0x01-0xff indicates an explicitly identified key.

Table 89—Elements of KeyDescriptor *(continued)*

Name	Type	Range	Description
Blacklisted	Boolean	TRUE or FALSE	Indicator as to whether the device previously communicated with this key prior to the exhaustion of the frame counter. If TRUE, this indicates that the device shall not use this key further, since it exhausted its use of the frame counter used with this key.
KeyDeviceList	List of KeyDeviceDescriptor entries (see Table 91)	–	A list of KeyDeviceDescriptor entries indicating which devices are currently using this key, including their blacklist status.
KeyDeviceListEntries	Integer	Implementation specific	The number of entries in KeyDeviceList.
KeyUsageList	List of KeyUsageDescriptor entries (see Table 90)	–	A list of KeyUsageDescriptor entries indicating which frame types this key may be used with.
KeyUsageListEntries	Integer	Implementation specific	The number of entries in KeyUsageList.
Key	Set of 16 octets	–	The actual value of the key.

Table 90—Elements of KeyUsageDescriptor

Name	Type	Range	Description
FrameType	Integer	0x00–0x03	See 7.2.1.1.1
CommandFrameIdentifier	Integer	0x00–0x09	See Table 82

Table 91—Elements of KeyDeviceDescriptor

Name	Type	Range	Description
DeviceAddress	IEEE address	Any valid 64-bit device address	The 64-bit IEEE extended address of the device in this KeyDeviceDescriptor.
Blacklisted	Boolean	TRUE or FALSE	Indicator as to whether the device indicated by DeviceAddress previously communicated with this key prior to the exhaustion of the frame counter. If TRUE, this indicates that the device shall not use this key further, since it exhausted its use of the frame counter used with this key.

7.6.2 Auxiliary security header

The auxiliary security header field has a variable length and contains information required for security processing, including a security control field, a frame counter field, and a key identifier field. The auxiliary

Table 92—Elements of SecurityLevelDescriptor

Name	Type	Range	Description
FrameType	Integer	0x00–0x03	See 7.2.1.1.1
CommandFrameIdentifier	Integer	0x00–0x09	See Table 82
SecurityLevelList	List of SecurityModeDescriptor entries (see Table 90)	–	A list of SecurityModeDescriptor entries indicating the security levels incoming MAC frames with the indicated frame type and, if present, command frame type are expected to be be secured with.
SecurityLevelListEntries	Integer	Implementation specific	The number of entries in SecurityLevelList.
DeviceOverrideSecurityMinimum	Boolean	TRUE or FALSE	Indicator as to whether originating devices for which the Exempt flag is set may override the required/expected security levels indicated by the SecurityLevelList element. If TRUE, this indicates that for originating devices with Exempt status, the incoming security level zero is acceptable, in addition to those incoming security levels indicated by the SecurityLevelList element.

Table 93—Elements of DeviceDescriptor

Name	Type	Range	Description
PANId	Device PAN ID	0x0000–0xffff	The 16-bit PAN identifier of the device in this DeviceDescriptor.
ShortAddress	Device short address	0x0000–0xffff	The 16-bit short address of the device in this DeviceDescriptor. A value of 0xfffe indicates that this device is only using its extended address. A value of 0xffff indicates that this value is unknown.
ExtAddress	IEEE address	Any valid 64-bit device address	The 64-bit IEEE extended address of the device in this DeviceDescriptor. This element is also used in unsecuring operations on incoming frames.
FrameCounter	Integer	0x00000000–0xffffffff	The incoming frame counter of the device in this DeviceDescriptor. This value is used to ensure sequential freshness of frames.
Exempt	Boolean	TRUE or FALSE	Indicator as to whether the device may override the minimum security level settings defined in Table 92.

security header field shall only be present if the security enabled subfield of the frame control field is set to one. The auxiliary security header field shall be formatted as illustrated in Figure 74.

Table 94—Elements of KeySourceDescriptor

Name	Type	Range	Description
ExtKeySource	Set of 8 octets	–	The 64-bit identifier of the key source (see 7.6.2.4.1).
ShortKeySource	Set of 4 octets	–	The 32-bit identifier of the ExtKeySource in this KeySourceDescriptor. A value of 0xfffffffe indicates that only ExtKeySource is used. A value of 0xffffffff indicates that this value is unknown.

Table 95—Elements of SecurityModeDescriptor

Name	Type	Range	Description
SecurityLevel	Integer	0x00–0x07	Security level identifier (see Table 96).

Octets: 1	4	0/1/5/9
Security control	Frame counter	Key identifier

Figure 74—Format of the auxiliary security header

7.6.2.1 Integer and octet representation

The auxiliary security header is a MAC frame field (see 7.2.1.7) and, therefore, uses the representation conventions specified in 7.2.

7.6.2.2 Security control field

The security control field is 1 octet in length and is used to provide information as to what protection is applied to the frame. The security control field shall be formatted as shown in Figure 75.

Bit: 0-2	3-4	5-7
Security level	Key identifier mode	reserved

Figure 75—Security control field format

7.6.2.2.1 Security level subfield

The security level subfield is 3 bits in length and indicates the actual frame protection that is provided. This value can be adapted on a frame-by-frame basis and allows for varying levels of data authenticity (to allow minimization of security overhead in transmitted frames where required) and for optional data confidentiality. The cryptographic protection offered by the various security levels is shown in Table 96. When nontrivial protection is required, replay protection is always provided.

Security levels can be ordered according to the corresponding cryptographic protection offered. Here, a first security level SEC1 is greater than or equal to a second security level SEC2 if and only if SEC1 offers at least the protection offered by SEC2, both with respect to data confidentiality and with respect to data authenticity. The statement, “SEC1 is greater than or equal to SEC2” shall be evaluated as TRUE if both of the following conditions apply:

a) Bit position b2 in SEC1 is greater than or equal to bit position b2 in SEC2 (where Encryption OFF < Encryption ON).
b) The integer value of bit positions b1 b0 in SEC1 is greater than or equal to the integer value of bit positions b1 b0 in SEC2 (where increasing integer values indicate increasing levels of data authenticity provided, i.e., MIC-0 < MIC-32 < MIC-64 < MIC-128).

Otherwise, the statement shall be evaluated as FALSE.

For example, ENC-MIC-64 >= MIC-64 is TRUE, since ENC-MIC-64 offers the same data authenticity protection as MIC-64, plus confidentiality. On the other hand, MIC-128 >= ENC-MIC-64 is FALSE, because even though MIC-128 offers stronger data authenticity than ENC-MIC-64, it offers no confidentiality.

Table 96—Security levels available to the MAC sublayer

Security level identifier	Security control field (Figure 75) b2 b1 b0	Security attributes	Data confidentiality	Data authenticity (including length M of authentication tag, in octets)
0x00	'000'	None	OFF	NO (M = 0)
0x01	'001'	MIC-32	OFF	YES (M=4)
0x02	'010'	MIC-64	OFF	YES (M=8)
0x03	'011'	MIC-128	OFF	YES (M=16)
0x04	'100'	ENC	ON	NO (M = 0)
0x05	'101'	ENC-MIC-32	ON	YES (M=4)
0x06	'110'	ENC-MIC-64	ON	YES (M=8)
0x07	'111'	ENC-MIC-128	ON	YES (M=16)

7.6.2.2.2 Key identifier mode subfield

The key identifier mode subfield is 2 bits in length and indicates whether or not the key that is used to protect the frame can be derived implicitly or explicitly; furthermore, it is used to indicate the particular representations of the key identifier field (see 7.6.2.4) if derived explicitly. The key identifier mode subfield shall be set to one of the values listed in Table 97. The key identifier field of the auxiliary security header (see 7.6.2.4) shall only be present if this subfield has a value that is not equal to 0x00.

7.6.2.3 Frame counter field

The frame counter field is 4 octets in length and represents the *macFrameCounter* attribute of the originator of a protected frame. It is used to provide semantic security of the cryptographic mechanism used to protect a frame and to offer replay protection.

Table 97—Values of the key identifier mode

Key identifier mode	Key identifier mode subfield b1 b0	Description	Key identifier field length (octets)
0x00	‘00’	Key is determined implicitly from the originator and receipient(s) of the frame, as indicated in the frame header.	0
0x01	‘01’	Key is determined from the 1-octet key index subfield of the key identifier field of the auxiliary security header in conjunction with *macDefaultKeySource*.	1
0x02	‘10’	Key is determined explicitly from the 4-octet key source subfield and the 1-octet key index subfield of the key identifier field of the auxiliary security header.	5
0x03	‘11’	Key is determined explicitly from the 8-octet key source subfield and the 1-octet key index subfield of the key identifier field of the auxiliary security header.	9

7.6.2.4 Key identifier field

The key identifier field has a variable length and identifies the key that is used for cryptographic protection of outgoing frames, either explicitly or in conjunction with implicitly defined side information. The key identifier field shall only be present if the key identifier mode subfield of the security control field of the auxiliary security header (see 7.6.2.2.2) is set to a value different from 0x00. The key identifier field shall be formatted as illustrated in Figure 76.

Octets: 0/4/8	1
Key source	Key index

Figure 76—Format for the key identifier field, if present

7.6.2.4.1 Key source subfield

The key source subfield, when present, is either 4 octets or 8 octets in length, according to the value specified by the key identifier mode subfield of the security control field (see 7.6.2.2.2), and indicates the originator of a group key.

7.6.2.4.2 Key index subfield

The key index subfield is 1 octet in length and allows unique identification of different keys with the same originator.

It is the responsibility of each key originator to make sure that actively used keys that it issues have distinct key indices and that the key indices are all different from 0x00.

7.6.3 Security operations

This clause describes the parameters for the CCM* security operations, as specified in B.2.2.

7.6.3.1 Integer and octet representation

This clause uses the representation conventions specified in B.1.

7.6.3.2 CCM* Nonce

The CCM* nonce is a 13-octet string and is used for the AES-CCM* mode of operation (see B2.2). The nonce shall be formatted as shown in Figure 77, with the leftmost field in the figure defining the first (and leftmost) octets and the rightmost field defining the last (and rightmost) octet of the nonce.

Octets: 8	**4**	**1**
Source address	Frame counter	Security level

Figure 77—CCM* nonce

The source address shall be set to the extended address *aExtendedAddress* of the device originating the frame, the frame counter to the value of the respective field in the auxiliary security header (see 7.6.2), and the security level to the security level identifier corresponding to the security level subfield of the security control field of the auxiliary security header as defined in Table 96.

The source address, frame counter and security level shall be represented as specified in 7.6.3.1.

7.6.3.3 CCM* prerequisites

Securing a frame involves the use of the CCM* mode encryption and authentication transformation, as described in B.3.1. Unsecuring a frame involves the use of the CCM* decryption and authentication checking process, as described in B.3.2. The prerequisites for the CCM* forward and inverse transformations are as follows:

— The underlying block cipher shall be the AES encryption algorithm as specified in B.2.1.
— The bit ordering shall be as defined in 7.6.3.1.
— The length in octets of the length field L shall be 2 octets.
— The length of the authentication field M shall be 0 octets, 4 octets, 8 octets, or 16 octets as required.

7.6.3.3.1 Authentication field length

The length of the authentication field M for the CCM* forward transformation and the CCM* inverse transformation is determined from Table 96, using the security level subfield of the security control field of the auxiliary security header of the frame.

7.6.3.4 CCM* transformation data representation

This clause describes how the inputs and output of the CCM* forward transformation, as described in B.3.1, are formed:

The inputs are

— Key
— Nonce
— *a* data
— *m* data

The output is

— *c* data

7.6.3.4.1 Key and nonce data inputs

The Key data for the CCM* forward transformation is passed by the outgoing frame security procedure described in 7.5.8.2.1. The Nonce data for the CCM* transformation is constructed as described in 7.6.3.2.

7.6.3.4.2 *a* data and *m* data

In the CCM* transformation process, the data fields shall be applied as in Table 98.

Table 98—*a* data and *m* data for all security levels

Security level identifier	*a* data	*m* data
0x00	None	None
0x01	MHR || Auxiliary security header || Non-payload fields || Unsecured payload fields	None
0x02	MHR || Auxiliary security header || Non-payload fields || Unsecured payload fields	None
0x03	MHR || Auxiliary security header || Non-payload fields || Unsecured payload fields	None
0x04	None	Unsecured Payload fields
0x05	MHR || Auxiliary security header || Non-payload fields	Unsecured Payload fields
0x06	MHR || Auxiliary security header || Non-payload fields	Unsecured Payload fields
0x07	MHR || Auxiliary security header || Non-payload fields	Unsecured Payload fields

7.6.3.4.3 *c* data output

In the CCM* transformation process, the data fields that are applied, or right-concatenated and applied, represent octet strings.

The secured payload fields right-concatenated with the authentication tag shall substitute the unsecured payload field in the original unsecured frame to form the secured frame (see Table 99).

7.6.3.5 CCM* inverse transformation data representation

This clause describes how the inputs and output of the CCM* inverse transformation, as described in B.3.2, are formed.

The inputs are

Table 99—c data for all security levels

Security level identifier	c data
0x00	None
0x01	MIC-32
0x02	MIC-64
0x03	MIC-128
0x04	Secured Payload fields
0x05	Secured Payload fields || MIC-32
0x06	Secured Payload fields || MIC-64
0x07	Secured Payload fields || MIC-128

— Key
— Nonce
— *c* data
— *a* data

The output is

— *m* data

7.6.3.5.1 Key and nonce data inputs

The Key data for the CCM* inverse transformation is passed by the incoming frame security procedure described in 7.5.8.2.3. The Nonce data for the CCM* transformation is constructed as described in 7.6.3.2.

7.6.3.5.2 *c* data and *a* data

In the CCM* inverse transformation process, the data fields shall be applied as in Table 100.

Table 100—*c* data and *a* data for all security levels

Security level identifier	*c* data	*a* data
0x00	None	None
0x01	MIC-32	MHR || Auxiliary security header || Non-payload fields ||Secured payload fields
0x02	MIC-64	MHR || Auxiliary security header || Non-payload fields || Secured payload fields
0x03	MIC-128	MHR || Auxiliary security header || Non-payload fields || Secured payload fields

Table 100—*c* data and *a* data for all security levels

Security level identifier	*c* data	*a* data
0x04	Secured payload fields	MHR || Auxiliary security header || Non-payload fields
0x05	Secured payload fields || MIC-32	MHR || Auxiliary security header || Non-payload fields
0x06	Secured payload fields || MIC-64	MHR || Auxiliary security header || Non-payload fields
0x07	Secured payload fields || MIC-128	MHR || Auxiliary security header || Non-payload fields

7.6.3.5.3 *m* data output

The *m* data shall then substitute secured payload fields and authentication tag in the original secured frame to form the unsecured frame.

7.7 Message sequence charts illustrating MAC-PHY interaction

This subclause illustrates the main tasks specified in IEEE P802.15.4REVb/D6. Each task is described by use of a message sequence chart to illustrate the chronological order, rather than the exact timing, of the primitives required for each task.

The primitives necessary for the PAN coordinator to start a new PAN are shown in Figure 78. The first action the next higher layer takes after resetting the MAC sublayer is to initiate a scan to search for other PANs in the area. An active scan is required, and an ED scan may optionally be performed. The steps for performing an active scan and an ED scan are shown in Figure 83 and Figure 79, respectively.

Once a new PAN is established, the PAN coordinator is ready to accept requests from other devices to join the PAN. Figure 80 shows the primitives issued by a device requesting association, while Figure 81 illustrates the steps taken by a coordinator allowing association. In the process of joining a PAN, the device requesting association will perform either a passive or an active scan to determine which PANs in the area are allowing association; Figure 82 and Figure 83 detail the primitives necessary to complete a passive scan and an active scan, respectively.

The primitives necessary for transmitting and receiving a single data packet are shown next. The actions taken by the originator of the packet are shown in Figure 84, while the actions taken by the recipient are shown in Figure 85.

When a device becomes unable to communicate to its coordinator any longer, the device can use an orphan scan to rediscover its coordinator. The primitives necessary for the realignment of an orphaned device are shown in Figure 86.