IEEE P802.15
Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)		
Title	**Security Functional Description from P802.15.4-REVc-DF5**		
Date Submitted	["17 March, 2015"]		
Source	[Pat Kinney] [Kinney Consulting]	Voice: E-mail:	[+1.847.960.3715] [pat.kinney@kinneyconsultingllc.com]
Re:	[This is an excerpt of the Security Functional Description from the latest 802.15.4 draft]		
Abstract	[Sub-clause 9.2.]		
Purpose	[The purpose of this document is to enhance discussion on the security of 802.15.45.]		
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.		

9. Security

9.1 Overview

The MAC sublayer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. This standard supports the following security services:

— Data confidentiality
— Data authenticity
— Replay protection

9.2 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 only if 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 9.2.1 and 9.2.3, respectively.

Security specific MAC PIB values are italicized and have a prefix of *sec*, i.e., *secKeyFrameCounter*.

Informative diagrams of the various security state machines can be found in Kivinen [B13].

9.2.1 Outgoing frame security procedure

The inputs to this procedure are the frame to be secured and the SecurityLevel, KeyIdMode, KeySource, and KeyIndex parameters. If the frame was generated in response to an MLME or MCPS primitive, then the value of SecurityLevel, KeyIdMode, KeySource and KeyIndex are set to the corresponding values of the primitive parameters. Otherwise, the inputs are:

— SecurityLevel shall be set to *macAutoRequestSecurityLevel*
— KeyIdMode shall be set to *macAutoRequestKeyIdMode*
— KeySource shall be set to *macAutoRequestKeySource*
— KeyIndex shall be set to *macAutoRequestKeyIndex*

The outputs from this procedure are the Status of the procedure and, if this Status is SUCCESS, the secured frame.

This procedure involves the following steps:

a) **Is security needed?** If the SecurityLevel parameter is zero, the procedure shall set the secured frame to be the frame to be secured and return with a Status of SUCCESS.
b) **Is security enabled?** If *macSecurityEnabled* is set to FALSE, the procedure shall return with a Status of UNSUPPORTED_SECURITY.
c) **Obtain KeyDescriptor.** The procedure shall obtain the KeyDescriptor using the KeyDescriptor lookup procedure as described in 9.2.2 with the DeviceAddressingMode set to Destination Addressing Mode field, DevicePanId set to the Destination PAN ID field, and DeviceAddress set to the Des-

tination Address field. If that procedure fails, the procedure shall return with a Status of UNAVAILABLE_KEY.

d) **Check frame counter value.**
   1) If TSCH mode is not being used and the *secFrameCounterPerKey* in the *secKeyDescriptor* is set to FALSE and *macFrameCounter* has the value 0xffffffff, the procedure shall return with a Status of COUNTER_ERROR.
   2) If TSCH mode is not being used and the *secFrameCounterPerKey* in the *secKeyDescriptor* is set to TRUE and *secKeyFrameCounter* has the value 0xffffffff, the procedure shall return with a Status of COUNTER_ERROR.

e) **Insert Auxiliary Security Header field.** The procedure shall insert the Auxiliary Security Header field in the frame to be secured, with the fields set as follows:
   1) The Security Level field of the Security Control field shall be set to the SecurityLevel parameter.
   2) The Key Identifier Mode field of the Security Control field shall be set to the KeyIdMode parameter.
   3) If TSCH mode is being used, the Frame Counter Suppression field in the Security Control field shall be set to one. Otherwise, the Frame Counter Suppression field in the Security Control field shall be set to zero.
   4) The Frame Counter field shall be set as follows:
      i) If TSCH mode is being used, the Frame Counter field shall be elided.
      ii) If the *secFrameCounterPerKey* in the *secKeyDescriptor* is set to TRUE, the Frame Counter field shall be set to *secKeyFrameCounter*.
      iii) Otherwise, the Frame Counter field shall be set to *macFrameCounter*.
   5) If the KeyIdMode parameter is set to a value not equal to zero, the Key Source and Key Index fields of the Key Identifier field shall be set to the KeySource and KeyIndex parameters, respectively.

f) **Secure the frame.** For the frames specified in Table 147, the Private Payload field and Open Payload field shall be set as indicated in the table. For frames not specified in Table 147, the Private Payload shall be set to the MAC Payload field and Open Payload field shall be empty. The procedure shall then use the Private Payload field, the Open Payload field, the *macExtendedAddress*, the Frame Counter field (if TSCH is not being used), the ASN (if TSCH is being used), the SecurityLevel parameter, and the *secKey* element of the KeyDescriptor to produce the secured frame according to the CCM* transformation process defined in 9.3.4.

Table 147—Exceptions to Private Payload field and Open Payload field definitions

Frame type	Private Payload field	Open Payload field
Beacon (Frame Version < 2)	Beacon Payload field	All other fields in the MAC Payload field
MAC Command (Frame Version < 2)	Content field	Command Identifier field

g) **Store frame counter.**
   1) If not using TSCH mode and *secFrameCounterPerKey* in the *secKeyDescriptor* is set to TRUE, the procedure shall increment *secKeyFrameCounter* by one.
   2) If not using TSCH mode and *secFrameCounterPerKey* in the *secKeyDescriptor* is set to FALSE, the procedure shall increment *macFrameCounter* by one.

Copyright © 2014 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

h) **Finish procedure.** The procedure shall return with a Status of SUCCESS.

9.2.2 KeyDescriptor lookup procedure

The inputs to this procedure are the KeyIdMode, KeySource, KeyIndex, DeviceAddressingMode, DevicePanId, and DeviceAddress. The outputs from this procedure are a Status and, if Status is set to SUCCESS, a KeyDescriptor.

This procedure involves the following steps:

a) If the KeyIdMode parameter is set to 0x00, then for each *secKeyIdLookupDescriptor* with *secKeyIdMode* set to 0x00 in the *macKeyIdLookupList*:
   1) If the DeviceAddressingMode is set to NONE, then the DevicePanId shall be set to *macPanId*. Otherwise, the DevicePanId shall be the value passed to the procedure.
   2) If the DeviceAddressingMode is set to NONE and the frame type is beacon, then the DeviceAddress shall be *macCoordExtendedAddress*.
   3) If the DeviceAddressingMode is set to NONE and the frame type is not beacon, then:
      i) If the *macCoordShortAddress* attribute is set to 0xfffe, then the DeviceAddress shall be set to the *macCoordExtendedAddress*.
      ii) If the *macCoordShortAddress* attribute is set to a value of 0x0000–0xfffd, then the DeviceAddress shall be set to the *macCoordShortAddress*.
      iii) If the *macCoordShortAddress* attribute is set to 0xffff, the procedure shall return with Status set to FAILED.
   4) If the DeviceAddressingMode is set to SHORT or EXTENDED, then the DeviceAddress shall be the value passed to the procedure.
   5) If the DeviceAddressingMode, DevicePanId, and DeviceAddress match the *secKeyDeviceAddrMode*, *secKeyDevicePanId*, and *secKeyDeviceAddress* of a *secKeyIdLookupDescriptor*, then the procedure returns with the corresponding *secKeyDescriptor* and Status set to SUCCESS.

b) If the KeyIdMode parameter is set to 0x01 and KeyIndex matches the *secKeyIndex* of a *secKeyIdLookupDescriptor* that has *secKeyIdMode* set to 0x01, then the procedure returns with the KeyDescriptor set to the corresponding *secKeyDescriptor* and Status set to SUCCESS.

c) If the KeyIdMode parameter is set to 0x02 or 0x03 and KeySource, KeyIdMode and KeyIndex match *secKeySource*, *secKeyIdMode*, and *secKeyIndex*, respectively, of a *secKeyIdLookupDescriptor*, then the procedure returns with the KeyDescriptor set to the corresponding *secKeyDescriptor* and Status set to SUCCESS.

d) The procedure shall return with Status set to FAILED.

NOTE—For broadcast frames, the KeyDescriptor lookup procedure will result in Status set to FAILED if implicit key identification is used. Hence, explicit key identification should be used for broadcast frames.[12]

9.2.3 Incoming frame security procedure, Security Enabled field is set to one

This procedure shall only used for incoming frames in which the Security Enabled field is set to one. For frames in which the Security Enabled field is set to zero, the procedure in 9.2.4 is used instead.

The input to this procedure is the frame to be unsecured. The outputs from this procedure are the Status of the procedure, the unsecured frame (including all IEs), SecurityLevel, KeyIdentifierMode, KeySource, KeyIndex and IeStatusList. The status for an IE in the IeStatusList is set to PASSED if the IE conforms to the security policy for that IE and is set to FAILED otherwise.

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

This is an unapproved IEEE Standards Draft, subject to change.

All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure.

This procedure involves the following steps:

a) **Legacy security**. If the Frame Version field of the frame to be unsecured is set to zero, the procedure shall return with a Status of UNSUPPORTED_LEGACY.

b) **Check for *macSecurityEnabled*.** If *macSecurityEnabled* is set to FALSE, the procedure shall return with a Status of UNSUPPORTED_SECURITY.

c) **Parse Auxiliary Security Header field.** the procedure shall set SecurityLevel and KeyIdentifierMode to the Security Level field and Key Identifier Mode field, respectively, of the frame to be unsecured. If required by the KeyIdentifierMode, the KeySource and KeyIndex shall be set to the Key Source field and Key Index field, respectively, of the Key Identifier field of the frame to be unsecured. If the resulting SecurityLevel is zero, the procedure shall return with a Status of UNSUPPORTED_SECURITY.

d) **Obtain source address.** DevicePanId shall be set to the Source PAN ID field, if it is present. Otherwise, DevicePanId shall be set to the Destination PAN ID field, if present. If neither PAN ID field is present, then DevicePanId shall be set to *macPanId*. The DeviceAddressingMode shall be set based on the Source Addressing Mode field using the mapping in Table 148. The DeviceAddress shall be set to the Source Address field, if present.

Table 148—Mapping of Source Addressing Mode field to DeviceAddressingMode

Source Addressing Mode field	DeviceAddressingMode
0x00	NONE
0x02	SHORT
0x03	EXTENDED

e) **Obtain KeyDescriptor.** The procedure shall obtain the KeyDescriptor using the KeyDescriptor lookup procedure as described in 9.2.2 with using the KeyIdMode, KeyIndex, KeySource, DeviceAddressingMode, DevicePanId, and DeviceAddress. If KeyDescriptor lookup procedure fails, the procedure shall return with a Status of UNAVAILABLE_KEY.

f) **Obtain DeviceDescriptor.** The procedure shall obtain the DeviceDescriptor using the DeviceDescriptor lookup procedure described in 9.2.5. If that procedure fails, then the procedure shall return with a Status of UNAVAILABLE_DEVICE.

g) **Obtain frame counter.** If TSCH mode is being used, then this step is skipped.

   1) If *secFrameCounterPerKey* of the KeyDescriptor is FALSE, the FrameCounterCheck value shall be set to be the *secDeviceFrameCounter* element of the DeviceDescriptor.

   2) If *secFrameCounterPerKey* of the KeyDescriptor is TRUE and there is a *secKeyDeviceFrameCounter* in the *secKeyDeviceFrameCounterList* in which *secDeviceExtAddress* matches *secExtAddress* of the DeviceDescriptor, then the procedure shall set the FrameCounterCheck value to the *secDeviceFrameCounter* of that *secKeyDeviceFrameCounter*.

   3) Otherwise, the procedure shall return with a Status of UNAVAILABLE_DEVICE.

h) **Check frame counter.** If TSCH mode is being used, then this step is skipped. If the Frame Counter field of the frame to be unsecured has the value 0xffffffff or the Frame Counter field of the frame to be unsecured is less than the FrameCounterCheck value, the procedure shall return with a Status of COUNTER_ERROR.

Copyright © 2014 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

i) **Unsecure frame.** For frames specified in Table 147, the Private Payload field and Open Payload field shall be set as indicated in the table. Otherwise, the Private Payload field shall be set to the MAC payload field and the Open Payload field shall be empty. The procedure shall then use the Private Payload field, the Open Payload field, *secExtAddress* of the DeviceDescriptor, the Frame Counter field of the frame to be unsecured, SecurityLevel, and *secKey* of the KeyDescriptor to produce the unsecured frame, according to the CCM* inverse transformation process described in the security operations, as described in 9.3.5. If the CCM* inverse transformation process fails, the procedure shall return with a Status of SECURITY_ERROR.

j) **Store frame counter.** If not using TSCH mode and *secFrameCounterPerKey* of the KeyDescriptor is FALSE, then *secDeviceFrameCounter* of the DeviceDescriptor shall be set to the value of the Frame Counter plus one. If not using TSCH mode and *secFrameCounterPerKey* of the KeyDescriptor is TRUE, then *secDeviceFrameCounter* of the *secKeyDeviceFrameCounter* corresponding to *secExtAddress* shall be set to the value Frame Counter plus one.

k) **Obtain SecurityLevelDescriptor.** The procedure shall obtain the SecurityLevelDescriptor by passing the Frame Type field and, if the frame is a MAC command, the Command Identifier field, of the frame to be unsecured to the SecurityLevelDescriptor lookup procedure described in 9.2.6. If that procedure fails, the procedure shall return with a Status of UNAVAILABLE_SECURITY_LEVEL.

l) **Check IE security.** If the IE present field of the frame to be unsecured is set to one, the procedure shall determine whether the IEs in the frame to be unsecured conforms to the security level policy by passing the DeviceDescriptor, SecurityLevelDescriptor and the SecurityLevel to the incoming IE security level checking procedure, as described in 9.2.7.

m) **Check IE Key Usage Policy.** If the IE present field of the frame to be unsecured is set to one, the procedure shall determine whether the frame to be unsecured conforms to the key usage policy by passing the IeStatusList, KeyDescriptor, the IEs from the frame, the Frame Type field, and, if the frame is a MAC command, the Command Identifier field, to the incoming IE key usage policy checking procedure, as described in 9.2.9.

n) **Check security level.** The procedure shall determine whether the frame to be unsecured conforms to the security level policy by passing the SecurityLevelDescriptor and the SecurityLevel to the incoming security level checking procedure, as described in 9.2.8. If that procedure returns with a Status of FAILED, the procedure shall return with the IEs from the unsecured frame, the IeStatusList and a Status of IMPROPER_SECURITY_LEVEL.

o) **Check key usage policy.** The procedure shall determine whether the frame to be unsecured conforms to the key usage policy by passing the KeyDescriptor, the Frame Type field, and, if the frame is a MAC command, the Command Identifier field, to the incoming key usage policy checking procedure, as described in 9.2.9. If that procedure fails, the procedure shall return with the IEs from the unsecured frame, the IeStatusList and a Status of IMPROPER_KEY_TYPE.

p) **Return unsecured frame.** The procedure shall return with the unsecured frame, SecurityLevel, KeyIdentifierMode, KeySource, KeyIndex, IeStatusList, and a Status of SUCCESS.

9.2.4 Incoming frame security procedure, Security Enabled field is set to zero

This procedure shall only used for incoming frames in which the Security Enabled field set to zero. For frames in which the Security Level field is set to one, the procedure in 9.2.3 is used instead.

The input to this procedure is the frame to be validated. The outputs from this procedure are the Status of the procedure, the validated frame (including all IEs) and IeStatusList. The status for an IE in the IeStatusList is set to PASSED if the IE conforms to the security policy for that IE and is set to FAILED otherwise.

All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure.

This procedure involves the following steps:

a) **Check for *macSecurityEnabled***. If *macSecurityEnabled* is set to FALSE, the procedure shall set the validated frame to be the frame to be validated and return with a Status of SUCCESS.
b) **Obtain source address.** DevicePanId shall be set to the Source PAN ID field, if it is present. Otherwise, DevicePanId shall be set to the Destination PAN ID field, if present. If neither PAN ID field is present, then DevicePanId shall be set to *macPanId*. DeviceAddressingMode shall be set to the Source Addressing Mode field. The DeviceAddress shall be set to the Source Address, if present.
c) **Obtain DeviceDescriptor.** The procedure shall obtain the DeviceDescriptor using the DeviceDescriptor lookup procedure described in 9.2.5 with the SecurityLevel set to zero. If that procedure fails, then the procedure shall return with a Status of UNAVAILABLE_DEVICE.
d) **Obtain SecurityLevelDescriptor.** The procedure shall obtain the SecurityLevelDescriptor by passing the Frame Type field and, if the frame is a MAC command, the Command Identifier field, of the frame to be validated to the SecurityLevelDescriptor lookup procedure described in 9.2.6. If that procedure fails, the procedure shall return with a Status of UNAVAILABLE_SECURITY_LEVEL.
e) **Check IE security.** If the IE present field of the frame to be validated is set to one, the procedure shall determine whether the IEs in the frame to be validated conforms to the security level policy by passing the DeviceDescriptor, the SecurityLevelDescriptor, the SecurityLevel and the IEs from the frame to the incoming IE security level checking procedure, as described in 9.2.7. That procedure will return the IeStatusList.
f) **Check security level.** The procedure shall determine whether the frame to be validated conforms to the security level policy by passing the SecurityLevelDescriptor and the SecurityLevel set to zero to the incoming security level checking procedure, as described in 9.2.8. If incoming security level checking procedure procedure returns with a Status of FAILED, the procedure shall return with a Status of IMPROPER_SECURITY_LEVEL. If the incoming security level checking procedure returned with a Status of CONDITIONALLY_PASSED and the *secExempt* element of the DeviceDescriptor is set to FALSE, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL.
g) **Return frame.** The procedure shall set the validated frame to be the frame to be validated and return with the frame to be validated, IeStatusList, and a status of SUCCESS.

9.2.5 DeviceDescriptor lookup procedure

The inputs to this procedure are DeviceAddressingMode, the DevicePanId, and the DeviceAddress. The output from this procedure is a Status of either PASSED or FAILED, and, if PASSED, a DeviceDescriptor.

This procedure involves the following steps:

a) If the DeviceAddressingMode is set to NONE, then the DevicePanId shall be set to *macPanId*. Otherwise, the DevicePanId shall be the value passed to the procedure.
b) If the DeviceAddressingMode is set to NONE, then:
    1) If the *macCoordShortAddress* attribute is set to 0xfffe, then the DeviceAddress shall be set to the *macCoordExtendedAddress*.
    2) If the *macCoordShortAddress* attribute is set to a value of 0x0000–0xfffd, then the DeviceAddress shall be set to the *macCoordShortAddress*.
    3) If the *macCoordShortAddress* attribute is set to 0xffff, the procedure shall return with a Status set to FAILED.
c) If the DeviceAddressingMode is set to SHORT or EXTENDED, then the DeviceAddress shall be the value passed to the procedure.
d) For each DeviceDescriptor, if DevicePanId matches the *secPanId* and DeviceAddress matches *secShortAddress*, if the DeviceAddressingMode is set to SHORT, or the *secExtAddress*, if the DeviceAddressingMode is set to EXTENDED, then the procedure shall return with the corresponding DevicesDescriptor and Status set to PASSED.

Copyright © 2014 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

e) The procedure shall return with a Status set to FAILED.

9.2.6 SecurityLevelDescriptor lookup procedure

The inputs to this procedure are the Frame Type field and, if the frame is a MAC command, the Command Identifier field. The output from this procedure are a Status of either PASSED or FAILED, and, if PASSED, a SecurityLevelDescriptor.

This procedure involves the following steps:

a) For each SecurityLevelDescriptor in the *macSecurityLevelTable* attribute:
   1) If the frame type indicates that the frame is not a MAC command and the Frame Type field is equal to the *secFrameType* element of the *secSecurityLevelDescriptor*, the procedure shall return with the *secSecurityLevelDescriptor* and Status set to PASSED.
   2) If the Frame Type field indicates that the frame is a MAC command and the Frame Type is equal to the *secFrameType* element of the *secSecurityLevelDescriptor* and the Command Identifier field is equal to the *secCommandIdentifier* element of the *secSecurityLevelDescriptor*, the procedure shall return with the *secSecurityLevelDescriptor* and Status set to PASSED.

b) The procedure shall return with Status set to FAILED.

9.2.7 Incoming IE security level checking procedure

The inputs to this procedure are DeviceDescriptor, SecurityLevelDescriptor, SecurityLevel, and the IEs in the frame. The output from this procedure is an IeStatusList, in which each element in the list is set to either PASSED or FAILED for each IE.

A match is found if the *secIeType* of the *secIeSecurityLevelDescriptor* matches the type of the IE and the *secIeId* of the *secIeSecurityLevelDescriptor* entry matches the ID of the IE.

This procedure involves the following steps:

a) If *secIeSecurityLevelDescriptorList* of the SecurityLevelDescriptor is empty, then set the IeStatus in the IeStatusList to PASSED for each IE in the frame, and return IeStatusList.

b) Set the IeStatus in the IeStatusList to FAILED for each IE in the frame.

c) For each IE in the frame and for each *secIeSecurityLevelDescriptor* in *secIeSecurityLevelDescriptorList*, if the IE matches the *secIeSecurityLevelDescriptor* entry, then
   1) If *secAllowedSecurityLevels* of the *secIeSecurityLevelDescriptor* is empty, then the procedure shall compare the SecurityLevel (as SEC1) with the *secSecurityMinimum* of the *secIeSecurityLevelDescriptor* (as SEC2) according to the algorithm described in 9.4.1.1. If this comparison evaluates to TRUE, the procedure shall set the IeStatus in the IeStatusList for this IE to PASSED.
   2) If *secAllowedSecurityLevels* of the *secIeSecurityLevelDescriptor* is not empty, the procedure shall check whether the SecurityLevel is equal to any of the elements of the *secAllowedSecurityLevels* of the *secIeSecurityLevelDescriptor*. If this check is successful, the procedure shall set the IeStatus in the IeStatusList for this IE to PASSED.
   3) If the SecurityLevel is equal to 0x00 and *secDeviceOverrideSecurityMinimum* of the *secIeSecurityLevelDescriptor* is set to TRUE, and the *secExempt* of the DeviceDescriptor is set to TRUE, the procedure shall set the IeStatus in the IeStatusList for this IE to PASSED.

d) Return IeStatusList.

This is an unapproved IEEE Standards Draft, subject to change.

9.2.8 Incoming security level checking procedure

The inputs to this procedure are SecurityLevelDescriptor and SecurityLevel. The output from this procedure is Status set to one of PASSED, FAILED, or CONDITIONALLY_PASSED.

The incoming security level checking procedure involves the following steps:

a) If *secAllowedSecurityLevels* in SecurityLevelDescriptor is empty, then the procedure shall compare the SecurityLevel (as SEC1) with the *secSecurityMinimum* element of the SecurityLevelDescriptor (as SEC2) according to the algorithm described in 9.4.1.1. If this comparison evaluates to TRUE, the procedure shall return with Status set to PASSED.

b) If *secAllowedSecurityLevels* in SecurityLevelDescriptor is not empty, the procedure shall check whether the SecurityLevel is equal to any of the elements of the *secAllowedSecurityLevels* of the SecurityLevelDescriptor. If this check is successful, the procedure shall return with Status set to PASSED.

c) If SecurityLevel is equal to 0x00 and the *secDeviceOverrideSecurityMinimum* element of the SecurityLevelDescriptor is set to TRUE, the procedure shall return with Status set to CONDITIONALLY_PASSED.

d) The procedure shall return with Status set to FAILED.

9.2.9 Incoming key usage policy checking procedure

The inputs to this procedure are the KeyDescriptor, the Frame Type field, and the Command Identifier field. The output from this procedure is Status set to PASSED or FAILED.

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

a) For each *secKeyUsageDescriptor* in the *secKeyUsageList* of the KeyDescriptor:

   1) If the Frame Type field indicates that the frame is not a MAC command and the Frame Type field is equal to the *secFrameType* element of the *secKeyUsageDescriptor*, the procedure shall return with Status set to PASSED.

   2) If the Frame Type field indicates that the frame is a MAC command, the Frame Type is equal to the *secFrameType* element of the *secKeyUsageDescriptor*, and the Command Identifier field is equal to the *secCommandIdentifier* element of the *secKeyUsageDescriptor*, the procedure shall return with Status set to PASSED.

b) The procedure shall return with Status set to FAILED.

9.2.10 Incoming IE key usage policy checking procedure

The inputs to this procedure are KeyDescriptor, IeStatusList, Frame Type field, the IEs in the frame, and, if the frame is a MAC command, the Command Identifier field. The output from this procedure is an IeStatusList, in which each element in the list is set to either PASSED or FAILED for each IE.

A match is found for an IE if the *secIeType* of the *secKeyIeUsageDescriptor* matches the type of the IE and the *secIeId* of the *secKeyIeUsageDescriptor* entry matches the ID of the IE.

This procedure involves the following steps:

a) Find the *secKeyUsageDescriptor* entry for which the Frame Type field matches *secFrameType* and, if the frame is a MAC command, the Command Identifier field matches *secCommandIdentifier*. If a matching *secKeyUsageDescriptor* entry was not found, or if the *secKeyIeUsageList* is empty in the *secKeyUsageDescriptor* that was found, then return IeStatusList.

Copyright © 2014 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

b) For each IE in the frame, if the IE does not match any of the *secKeyIeUsageDescriptor* entries, then set the IeStatus in the IeStatusList for this IE to FAILED.

c) Return IeStatusList.

9.3 Security operations

This subclause describes the parameters for the CCM* security operations, as specified in B.3.2.

9.3.1 Integer and octet representation

The integer and octet representation conventions specified in B.2 are used throughout this subclause.

9.3.2 CCM* nonce

9.3.2.1 CCM* nonce for non-TSCH mode

The CCM* nonce for non-TSCH mode shall be formatted as shown in Figure 216, with the leftmost field in the figure defining the first octets and the rightmost field defining the last octet of the nonce.

Octets: 8	**4**	**1**
Source Address	Frame Counter	Nonce Security Level

Figure 216—CCM* nonce for non-TSCH mode

The Source Address field shall be set to the extended address of the device originating the frame.

The Frame Counter field shall be set to the value of the respective field in the Auxiliary Security Header field, as defined in 9.4.

The Nonce Security Level field is an unsigned integer that shall be set to the value of the Security Level field of the Security Control field, as defined in 9.4.1.

The Source Address field, Frame Counter field, and Security Level field shall be represented as specified in 9.3.1.

9.3.2.2 CCM* nonce for TSCH mode

When TSCH mode is enabled, the nonce shall be formatted as shown in Figure 217.

Octets: 8	**5**
Source Address	ASN

Figure 217—CCM* nonce in TSCH mode

The Source Address shall be set to the extended address of the device originating the frame.

The ASN shall be set to the ASN of the timeslot during which the frame is sent.

Copyright © 2014 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.